P.1877/88

5 1988

informatyka

Baza danych PROMIS

Procedury INTR i MSDOS w MS-Pascalu

WYDAWNICTWO NOT SIGMA

Cena 200 zł

informatyka

5 1988

Nr 5

Miesięcznik Rok XXIII

Maj 1988

Organ Komitetu Naukowo-Technicznego NOT ds. Informatyki

KOLEGIUM REDAKCYJNE:

Mgr Jarosław DEMINET, dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (redaktor naczelny), dr inż. Marek MACHURA, dr inż. Wiktor RZECZKOWSKI, mgr inż. Jan RYŻKO, mgr Hanna WŁODARSKA, dr inż. Janusz ZALEWSKI (zastępca redaktora naczelnego).

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Juliusz Lech KULIKOWSKI

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34.

Zakł. Graf. „Tamka". Zam. 0203-1300/88. Obj. 4,0 ark. druk. Nakład 8950 egz. U-3.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 200 zł
Prenumerata roczna 2400 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA CZASOPISM I KSIĄŻEK TECHNICZNYCH

SIGMA

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE

	Strona
Promis — system zarządzania bazą danych (1) *Halina Ciechomska, Teresa Wójciekian, Ewa Zaborowska*	1
Języki wysokiego poziomu do przetwarzania napisów — COMIT, SNOBOL4 i Icon (2) *Ralph Griswold, Madge T. Griswold*	4
Algorytmy kombinatoryjne i ich efektywność (3). Kolorowanie wierzchołków grafu *Maciej M. Sysło*	8
Procedury INTR i MSDOS w MS-Pascalu *Michał Bartyś*	12
Mazovia 1016 — konfiguracja sprzętowa *Krzysztof Dzik*	15
Turbo C. Zbiory nagłówkowe, prototypy, deklarowanie funkcji *Jan Bielecki*	17
Komputerowe przetwarzanie wyników badań medycznych *Henryk Biesiada, Jerzy Grzywocz*	19
Z KRAJU	23
INFOSYSTEM '88	
ZE ŚWIATA	24
Kompilator Ady dla mikroprocesora MC8000 Systemy ekspertowe — stan obecny i przyszłość Esperanto jako język pośredni w tłumaczeniu komputerowym	
TERMINOLOGIA	30
Pliki i zbiory	
NOWE KSIĄŻKI	31
Problematyka baz danych	

W NAJBLIŻSZYCH NUMERACH

- Jan Zabrodzki rozpoczyna cykl artykułów na temat cyfrowego generowania obrazów
- Andrzej Turowicz charakteryzuje lokalne sieci komputerowe D-Link
- Dariusz Caban omawia transputery — narzędzia przetwarzania równoległego
- Jan Bielecki przedstawia sposoby i przykłady użycia preprocesora oraz makrodefinicji w języku C
- Mariusz Kuc opisuje dwie wersje Turbo Pascala — dla systemów operacyjnych CP/M i PC-DOS oraz porównuje je z innymi kompilatorami Pascala

HALINA CIECHOMSKA
TERESA WOJCIEKIAN
EWA ZABOROWSKA
Instytut Maszyn Matematycznych
Warszawa

P.1817/88

Promis — system zarządzania bazą danych (I)

PROMIS jest systemem zarządzania bazą danych przeznaczonym dla minikomputerów SM-4 (i zgodnych z nimi) z systemem operacyjnym DOC PB lub RSX 11-M. Jest on adaptacją systemu ADABAS-M opracowanego dla minikomputerów PDP-11. Adaptacji tej dokonano w Instytucie Maszyn Matematycznych w Warszawie.

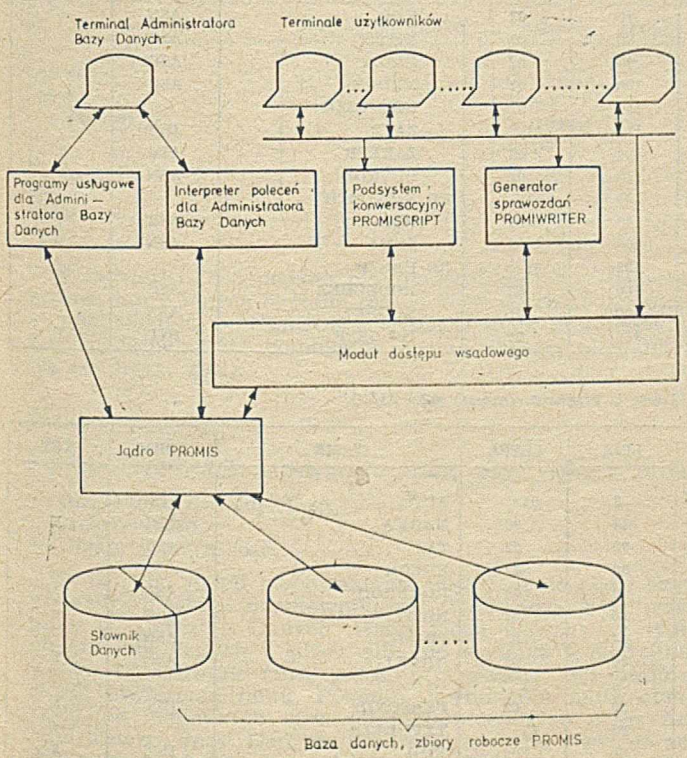
Najważniejsze cechy systemu PROMIS, decydujące o jego walorach użytkowych, to:

- wielodostępność do bazy danych,
- duża szybkość dostępu,
- mechanizmy ochrony danych,
- różnorodność form pracy z bazą danych, dzięki którym użytkownik może działać w sposób odpowiedni do swoich kwalifikacji i potrzeb.

Praca z systemem może się odbywać w następujących czterech trybach:

- konwersacyjnie, z użyciem uprzywilejowanych poleceń lub programów usługowych przeznaczonych dla administratora bazy danych (ABD),
- konwersacyjnie, z użyciem podsystemu PROMISCRIPT,
- przez odwołania typu CALL z programów napisanych w języku Fortran, Basic 2-Plus, Cobol lub Macro,
- przez program napisany w specjalnym języku PRW, interpretowanym przez podsystem PROMIWRITER.

W bieżącej części artykułu omówiono architekturę systemu oraz pracę w trybie konwersacyjnym przeznaczonym dla administratora bazy danych.



Architektura systemu PROMIS

CHARAKTERYSTYKA OGÓLNA

PROMIS pracuje pod kontrolą systemu operacyjnego DOC PB wersja 2.0 (lub późniejsze), w którym zaimplementowano programy obsługi urządzeń zewnętrznych i globalną sygnalizację zdarzeń (ang. event flags). Efektywna praca systemu wymaga następującej konfiguracji sprzętowej:

— procesor SM-2104 lub równoważny wyposażony w jednostkę zarządzania pamięcią (ang. Memory Management Unit),
— pamięć operacyjna o pojemności 128 K słów,
— pamięć dyskowa o pojemności 5 MB (plus obszar na bazę danych),
— pamięć taśmowa,
— dwa monitory ekranowe.

Zasadniczo system PROMIS umożliwia równoczesną pracę kilku użytkowników tylko z jedną bazą danych. Jednak w wypadku konfiguracji z dużymi zasobami pamięci operacyjnej (powyżej 128 K słów) można go przystosować do działania w warunkach równoczesnego korzystania wielu użytkowników z wielu baz danych (tryb MULTIDBMS).

Architekturę systemu PROMIS przedstawiono na rysunku. Centralną częścią systemu jest program zwany jądrem PROMIS. Program ten stale rezyduje w pamięci i jest odpowiedzialny za wielodostęp do bazy danych, obsługę wszystkich zbiorów pomocniczych systemu oraz realizację operacji na najniższym poziomie. Z usług tego programu korzystają wszystkie inne moduły systemu PROMIS. Inne elementy architektury, a także ich znaczenie dla współpracy użytkowników z bazą danych scharakteryzowano niżej.

BAZA DANYCH

Charakteryzując bazę danych PROMIS opiszemy:
— jak użytkownicy powinni wyobrażać sobie jej strukturę (struktura logiczna),
— jak baza jest zapisana w fizycznych zbiorach na dyskach (struktura fizyczna).

Trzeba przy tym zaznaczyć, że pełny obraz struktury logicznej bazy musi znać jej administrator, natomiast poszczególni użytkownicy mogą posługiwać się tylko jej fragmentami.

Struktura logiczna

Baza danych systemu PROMIS jest zbiorem maksymalnie 250 plików, z których każdy zawiera rekordy (do 16 milionów) o tej samej strukturze logicznej, właściwej danemu plikowi. W zależności od liczby rekordów pliki dzielą się na małe (do 65 K rekordów) i duże (powyżej 65 K rekordów). Każdy plik ma nazwę, niepowtarzalną w obrębie plików należących do tej samej bazy danych. Na etapie wprowadzania danych, rekordom są przypisywane wewnętrzne identyfikatory zwane **ISN** (ang. Internal Sequence Number).

Rekord może składać się z jednej lub wielu (do 250) pozycji, mających niepowtarzalne nazwy w ramach pliku. Pozycja może być:

- **prosta**, tj. zawierająca jedną wartość,
- **grupowa**, tj. stanowiąca grupę pozycji prostych związanych nadrzędną nazwą,
- **grupowa krotna**, czyli pozycja grupowa zawierająca więcej niż jeden zestaw wartości pozycji prostych.

PROMIS akceptuje następujące typy wartości:
— tekstowe w kodzie ASCII, do 128 znaków,
— tekstowe w kodzie ASCII, do 128 znaków,
— numeryczne w kodzie ASCII, do 20 znaków,
— tekstowe w kodzie RAD50, do 64 słów,
— daty w postaci zwartej, 6 znaków,
— daty z separatorami, 8 znaków,
— liczby całkowite dwójkowe, do 8 znaków.

W każdym pliku wybrane pozycje (do 32) mogą zostać zadeklarowane jako pozycje deskryptorowe (deskryptory). Deklaracja deskryptora umożliwia wyszukiwanie rekordów w pliku na podstawie wartości tego deskryptora (wartości te są nazywane kluczami). Wyszukiwanie bezpośrednio, według wartości występujących w pozycji nie zadeklarowanej jako deskryptor jest w systemie PROMIS niemożliwe. Jednakże na takie pozycje mogą być nakładane pewne dodatkowe warunki w języku PRW (język generatora sprawozdań PROMIWRITER).

Dostęp do rekordów danego pliku można zaklasyfikować jako:

- **bezpośredni** — kryterium wyszukiwania rekordów stanowią warunki nałożone na wartości jednego lub kilku deskryptorów, połączone operatorami logicznymi,
- **sekwencyjny według wybranego klucza** — przy wyszukiwaniu określa się początkową wartość deskryptora i otrzymuje ciąg rekordów uporządkowany według rosnących wartości tego deskryptora,
- **sekwencyjny według ISN** — przy wyszukiwaniu określa się początkową wartość wewnętrznego identyfikatora rekordu (ISN) i otrzymuje ciąg rekordów uporządkowany według rosnących wartości ISN.

Logiczną strukturę pliku zapisanego w bazie danych systemu PROMIS obrazuje schemat, czyli opis wszystkich pozycji tego pliku. Dla każdej pozycji pliku schemat zawiera m.in. jej nazwę i długość, typ oraz wskaźnik określający, czy ta pozycja jest deskryptorem i jakiego rodzaju (np. pierwotnym lub niezerowym). Informacje wyprowadzane z pliku na zewnątrz przez konkretnego użytkownika lub wprowadzane do pliku mogą mieć jednak trochę inną strukturę logiczną niż określona w schemacie. Opisy takich struktur zewnętrznych danego pliku nazywają się podschematami. Schemat i podschematy każdego pliku są przechowywane w Słowniku Danych.

Struktura fizyczna

Każdemu plikowi w bazie danych PROMIS są przydzielone dwa obszary fizyczne:

- **Obszar Danych** — zawierający wszystkie rekordy tego pliku,
- **Asocjator** — zawierający listy inwersyjne dla deskryptorów tego pliku oraz zbiór par (ISN, adres rekordu) zwany **Konwerterem Adresów**; służy on do wyznaczania fizycznych adresów rekordów na podstawie wewnętrznych identyfikatorów ISN.

Rekordy są przechowywane w **Obszarze Danych** jako ciągi bajtów o zmiennej długości (do 2000 bajtów). Każdy rekord zawiera przypisany mu przez system wewnętrzny identyfikator ISN oraz oddzielnie adresowalne wartości pozycji. Przed zapisaniem w bazie, wartości pozycji podlegają kompresji zgodnej z ich typem, przy czym wartości puste są pomijane. Każdy rekord zajmuje spójny obszar pamięci, ale może być rozmieszczony w kilku blokach; w jednym bloku system może umieścić kilka rekordów. Między rekordami w Obszarze Danych nie ma żadnych powiązań. Wolna przestrzeń powstała po usunięciu rekordów może być ponownie wykorzystana. Z tego powodu Obszar Danych nie wymaga okresowych reorganizacji.

Asocjator zawiera wszystkie struktury wykorzystywane przy dostępie do pliku. Dla każdego deskryptora w obszarze Asocjatora jest przechowywana lista inwersyjna, w której z poszczególnymi wartościami deskryptora są skojarzone listy wewnętrznych identyfikatorów rekordów zawierających te wartości. Fizyczny adres rekordu jest zawsze odnajdowany na podstawie ISN, z użyciem Konwertera Adresów. Dzięki temu w wypadku przeniesienia rekordu do innego miejsca pamięci aktualizacji podlega tylko jeden odpowiadający mu element Konwertera Adresów, niezależnie od liczby deskryptorów zdefiniowanych dla danego pliku. Wartości występujące w listach inwersyjnych, podobnie jak w Obszarze Danych, są pamiętane w postaci skróconej, uzyskanej w wyniku zastosowania kompresji, odpowiednio do ich typu. Przy odwołaniach do Asocjatora, PROMIS stosuje metodę dostępu, umożliwiającą odnalezienie całej poszukiwanej listy rekordów po jednokrotnym odwołaniu.

Podczas realizacji bezpośredniego dostępu do rekordów, PROMIS najpierw tworzy wynikową listę ISN, operując listami ISN odpowiadającymi wartościom deskryptorów podanych przez użytkownika w kryterium wyszukiwania, a następnie na podstawie tej listy, posługując się Konwerterem Adresów, wybiera odpowiednie rekordy. Przy dostępie sekwencyjnym według wskazanego klucza, tj. według wskazanej wartości doskryptora, PROMIS inicjuje przeglądanie odpowiednich list dla tego deskryptora w kolejności wzrastania wartości, zaczynając od wartości zadanej. Konwerter Adresów służy do odwzorowania wartości kolejnych znalezionych indentyfikatorów wewnętrznych na fizyczne adresy rekordów. Najefektywniej PROMIS realizuje dostęp sekwencyjny według ISN (niezależny od wartości danych w rekordach), gdyż w tym wypadku nie przegląda on żadnych list inwersyjnych, a korzysta wyłącznie z Konwertera Adresów.

Obsługa Asocjatora jest wewnętrzną sprawą systemu PROMIS. Wszelkie dopisywanie rekordów, skreślanie ich i zmiany zawartości pozycji, dokonywane przez użytkownika, implikują automatyczne i natychmiastowe zmiany odpowiednich list Asocjatora.

Przykład bazy danych

Wszelkie przykłady przytoczone w dalszej części artykułu dotyczą bazy danych złożonej z dwóch plików (kartotek):
— **kartoteki osobowej PERSON,**
— **kartoteki AUTO,** zawierającej dane dotyczące pojazdów.
W plikach tych zgromadzono dane pewnego przedsiębiorstwa ubezpieczeniowego. W tabelach 1 i 2 przedstawiono fragmenty schematów odpowiednio dla pliku PERSON i AUTO. Pominięto w nich informacje dotyczące długości pozycji, rodzaju deskryptorów oraz stopnia wypełnienia Aso-

Tabela 1. Fragment schematu pliku PERSON

LINE	LEVEL	NAME	FORM	KEY
10	02	PERSONID	INT	Y
15	01	DANEPERS		
20	02	NAZW	ASC	
30	02	IMIE	ASC	
40	02	DATAUR	ASC	
50	02	PLEC	ASC	
60	02	STCYW	ASC	Y
65	01	ADRES		
70	02	KOD	ASN	
80	02	MIEJSC	ASC	
90	02	ULICA	ASC	
100	02	NRDOM	ASC	
110	02	NRMIESZK		
115	01	ZATR	ASC	
120	02	ZAKLPR	ASC	Y
130	02	DOCHOD	ASN	
135	01	RG HISTDOCH		
140	02	ROK	ASC	Y
150	02	SUMA	ASN	Y
155	01	RG UBEZP		
160	02	RODZUBEZ	ASC	Y
170	02	WYSSKL	ASN	
180	02	IDUBEZP	INT	Y

Tabela 2. Fragment schematu pliku AUTO

LINE	LEVEL	NAME	FORM	KEY
5	01	AUTO		
10	02	MARKA	ASC	Y
20	02	TYP	ASC	
30	02	KOLOR	ASC	
40	02	ROKPROD	ASN	
50	02	NRSIL	ASN	
60	02	POJSIL	ASN	
70	02	NRREJ	ASC	Y
75	01	ID		
80	02	PERSONID	INT	Y
85	01	TYPUBEZP		
90	02	KOD	INT	Y

cjatora. Na schematach typu pozycji są oznaczone w następujący sposób:
01 — pozycja grupowa,
02 — pozycja prosta,
01 RG — pozycja grupowa krotna.
Pozycje, które są deskryptorami oznaczono literą Y.

W pliku PERSON znajdują się informacje o osobach ubezpieczonych, takie jak:

- identyfikator ubezpieczonego PERSONID,
- dane personalne DANEPERS (nazwisko NAZW, imię IMIĘ, data urodzenia DATUR, stan cywilny STCYW, płeć PLEC),
- adres ADRES (KOD, miejscowość MIEJSC, ULICA, numer domu NRDOM, numer mieszkania NRMIESZK),
- dane o zatrudnieniu ZATR (nazwa aktualnego zakładu pracy ZAKLPR, suma dochodów w roku poprzednim DOCHOD),
- historia dochodów HISTDOCH (ROK, wysokość dochodów SUMA),
- dane o rodzaju zawartych ubezpieczeń UBEZP (rodzaj ubezpieczenia RODZUBEZ, wysokość składki WYSSKL, identyfikator ubezpieczenia IDUBEZP).

W pliku AUTO zapisano informacje o ubezpieczonych pojazdach:

- dane opisujące pojazd AUTO (MARKA, TYP, KOLOR, rok produkcji ROKPROD, numer silnika NRSIL, pojemność silnika POJSIL, numer rejestracyjny NRREJ),
- dane dotyczące właściciela ID (identyfikator ubezpieczonego PERSONID),
- dane dotyczące typu ubezpieczenia TYPUBEZP (kod ubezpieczenia KOD).

Tabela 3. Przykłady podschematów dotyczących pliku PERSON

Podschemat PLOAD	Podschemat PDOCH	Podschemat PFULL
10 PERSONID	135 HISTDOCH 10	20 NAZW
20 DANEPERS		22 5SP
30 ADRES		30 IMIE
40 ZATR		32 1SP
50 HISTDOCH		40 PERSONID
60 UBEZP		45 1CR
		48 5SP
		50 RODZUBEZ

Tabela 4. Przykłady podschematów dotyczących pliku AUTO

Podschemat ALOAD	Podschemat AAUTO	Podschemat AFULL
10 AUTO	10 AUTO	10 MARKA
20 ID		20 3SP
30 TYPUBEZP		30 NRREJ
		40 5SP
		50 KOD
		60 3SP

W tabelach 3 i 4 podano przykładowe definicje podschematów, które można określić mając dany schemat dla pliku — odpowiednio — PERSON i AUTO (opis przeznaczenia tych podschematów pominięto ze względu na brak miejsca — przyp. red.).

Ochrona danych

System PROMIS zapewnia ochronę bazy danych przed:
— niepowołanym dostępem,
— zniszczeniem,
— utrata integralności.

Do ochrony przed niepowołanym dostępem do pliku służą hasła przydzielane użytkownikom i wprowadzane do opisu pliku w Słowniku Danych. Hasła mogą upoważniać tylko do czytania danych z pliku albo do czytania i modyfikowania. Użytkownik musi podać przydzielone mu hasło w chwili otwierania pliku. Ponadto, użytkownik może otworzyć plik dopiero po uprzednim otworzeniu go przez Administratora Bazy Danych i określeniu typu współpracy (czytanie, modyfikowanie, ładowanie danych).

Do ochrony bazy przed zniszczeniem służy:
— kronika systemu, tj. zbiór, w którym system odnotowuje wszelkie modyfikacje bazy danych,
— polecenia ABD, umożliwiające składowanie bazy danych i kroniki oraz odtwarzanie bazy po awarii na podstawie jej ostatniej kopii i kolejnych kopii kroniki.

Użyte mechanizmy ochronne danych zapewniają integralność bazy w sposób automatyczny, ponieważ nie dopuszczają równoczesnego wykonywania aktualizacji tego samego rekordu: każdą aktualizację musi poprzedzić zablokowanie danych na poziomie rekordu.

Słownik Danych

System PROMIS jest sterowany Słownikiem Danych. Oznacza to, że wszelkie kontakty użytkownika z bazą danych są realizowane na podstawie informacji ze Słownika Danych. Słownik dzieli się na dwie części:
— obszar definicji danych,
— obszar sterowania.

Obszar definicji danych zawiera informacje o wszystkich plikach, pozycjach i podschematach zdefiniowanych dla danej bazy. W szczególności dla każdego pliku w obszarze definicji danych jest określona charakterystyka pliku, sposób rozmieszczenia pliku na dyskach i hasła dostępu (dla czytania i modyfikowania). Informacje dotyczące poszczególnych pozycji obejmują: domniemany typ wartości i długość, rodzaj pozycji, ewentualną charakterystykę deskryptora. W odniesieniu do podschematów obszar definicji danych zawiera informacje o występujących w nich pozycjach, ich typach i długości, maksymalnej liczbie powtórzeń oraz parametrach określających postać zewnętrzną rekordu (np. dodatkowych tekstach, spacjach).

Obszar sterowania zawiera parametry eksploatacyjne ustalane przy starcie systemu PROMIS, takie jak:
— wielkość obszarów przeznaczonych na listy aktualnie wykorzystywanych nazw plików i podschematów,
— liczba kanałów dostępu, tj. obszarów wykorzystywanych przez mechanizm systemowy, umożliwiający jednoczesny dostęp do kilku plików lub do jednego pliku różnymi metodami (np. jednocześnie bezpośrednio i sekwencyjnie),
— lista plików otwieranych automatycznie przy starcie systemu.

POLECENIA I PROGRAMY USŁUGOWE DO DYSPOZYCJI ADMINISTRATORA BAZY DANYCH

Do obowiązków Administratora Bazy Danych ABD należy:
— zakładanie bazy danych,
— przygotowanie sesji współpracy użytkowników z systemem PROMIS,
— ochrona bazy przed zniszczeniem.

Administrator realizuje te zadania wywołując niezależne programy usługowe (poleceniem RUN), specjalne procedury MCR lub uprzywilejowane polecenia ABD.

Zakładanie bazy danych odbywa się w dwóch etapach:
1. Przydzielenie obszarów dyskowych dla Słownika Danych oraz dla zbiorów pomocniczych systemu PROMIS związanych z tą bazą (np. kronika zmian, zbiory robocze stosowane przy sortowaniu). Tę funkcję ABD wykonuje za pomocą specjalnego uprzywilejowanego programu usługowego PRMGEN, sterującego generowaniem odpowiedniej wersji systemu PROMIS, przystosowanej do współpracy z daną bazą.

2. Budowanie plików bazy polegające na przydzieleniu obszarów na dane i asocjatory poszczególnych plików, określenie haseł dostępu, zdefiniowanie ich schematu i podschematów, a następnie załadowanie danych do przygotowanych obszarów. Wymienione operacje realizują uprzywilejowane polecenia ABD.

Przygotowanie sesji współpracy systemu PROMIS z użytkownikami polega na ewentualnym zainstalowaniu zadań systemu (za pomocą specjalnej procedury MCR), zapoczątkowaniu zadania głównego (poleceniem RUN PROMIS) i otwarciu do czytania lub aktualizowania plików przeznaczonych do współpracy podczas danej sesji (za pomocą polecenia ABD).

dokończenie na s. 31

RALPH E. GRISWOLD
MADGE T. GRISWOLD
Department of Computer Science
Uniwersytet stanu Arizona, Tucson
USA

Języki wysokiego poziomu do przetwarzania napisów COMIT, SNOBOL4 i Icon (2)

W obecnej części artykułu omówiono SNOBOL4. Współczesna wersja tego języka jest wynikiem długiego rozwoju. Wersja najwcześniejsza, opracowana w 1962 roku w Bell Laboratories przez Davida J. Farbera, Ralpha E. Griswolda i Ivana P. Polonsky'ego, była dość prostym manipulatorem napisów, zaprojektowanym do przekształcania wzorców symbolicznych. Obecna wersja, SNOBOL4, zdefiniowana przez Griswolda, Polonsky'ego i Jamesa F. Poage'a, jest rozbudowanym językiem uniwersalnym, w którym położono nacisk na przetwarzanie zarówno napisów, jak i struktur listowych.

SNOBOL4 ma liczne właściwości umożliwiające programowanie w sposób łatwy i zwarty. Typy danych obejmują liczby całkowite, rzeczywiste, napisy, wzorce, tablice i tabele. W języku tym nie daklaruje się typów — ich sprawdzanie i ewentualne zamiany są wykonywane automatycznie. SNOBOL4 ma rozbudowany mechanizm dopasowywania wzorców tekstowych. Ponadto, język ten posiada wyrafinowane struktury (w kategorii danych), procedury rekurencyjne oraz możliwość dynamicznej kompilacji, która czyni wykonanie programu procesem niezwykle elastycznym. Wprowadzanie i wyprowadzanie jest wykonywane niejawnie podczas użycia stosownych zmiennych.

Działania podstawowe

Podstawowe operacje języka przypominają ich odpowiedniki w innych językach programowania.

Na przykład:

```
i = i + j
```

oznacza zwiększenie **i** o **j**;

```
average = sum / 12.0
```

oznacza przypisanie zmiennej **average** wartości zmiennej **sum** podzielonej przez **12.0**;

```
j = REMDR(i,3)
```

oznacza przypisanie zmiennej **j** reszty z dzielenia **i** przez 3. Funkcje takie jak **REMDR** zapisuje się wielkimi literami. Nazwy innych obiektów, np. **i** lub **sum**, mogą zawierać zarówno wielkie, jak i małe litery.

Ciągi znaków — **łańcuchy** [1]) — stanowią w języku SNOBOL4 osobną kategorię danych, czego nie należy mylić z reprezentacją napisów w językach pascalopodobnych, gdzie są one wyrażane za pomocą tablic znaków. Łańcuchy są „niepodzielne" w tym sensie, że nie istnieją działania zmieniające poszczególne znaki w łańcuchu. Niemniej, jedne łańcuchy mogą być otrzymywane z innych w drodze rozmaitych (całościowych) przekształceń.

[1]) Będziemy się posługiwać tą nazwą w celu odróżnienia specyficznej kategorii danych, jaką stanowią napisy w języku SNOBOL4 (przyp. tłum.).

Do literalnego przedstawienia w programie łańcuchów służą cudzysłowy, którymi wydziela się łańcuchy z reszty tekstu, np.:

```
salutation = "Hello there"
```

gdzie następuje przypisanie 11-znakowego łańcucha zmiennej **salutation** (odstęp między **s** i **l** jest równoprawnym znakiem, jak każdy inny). Łańcuch pusty, nie zawierający ani jednego znaku, jest reprezentowany przez dwa cudzysłowy napisane tuż obok siebie:

```
null = ""
```

Łańcuch może zawierać dowolny ze znaków należących do zestawu używanego na docelowym komputerze. Łańcuchy mogą być dowolnej długości; pamięć dla nich rezerwuje się automatycznie, stosownie do potrzeb.

Operacja spinania łańcuchów nie ma jawnej reprezentacji — jest powodowana przez napisanie łańcuchów obok siebie, oddzielonych odstępem. Przykładowo, jeśli zmienna **salutation** ma poprzednią wartość, to instrukcja:

```
remark = "[" salutation "!]"
```

spowoduje przypisanie łańcucha:

```
[Hello there!]
```

zmiennej **remark**.

Istnieją funkcje wytwarzające łańcuchy, np.:

```
banner — DUPL ("+—",60)
```

co powoduje przypisanie zmiennej **banner** wartości 60-krotnego powtórzenia łańcucha "+—". Istnieją również funkcje działające na łańcuchach, jak:

```
length = SIZE(banner)
```

gdzie zmiennej **length** nadaje się wartość równą liczbie znaków aktualnie pamiętanych przez **banner** (120).

Struktura programu i sterowanie

Program w języku SNOBOL4 składa się z ciągu instrukcji wykonywanych po kolei, chyba że określi się zmianę tej kolejności. Instrukcje mogą mieć etykiety i **pola skoków**. Etykietą — jeśli taka występuje — jest pierwszy element instrukcji. **Skoki** są poprzedzane dwukropkami i ujmowane w nawiasy. Wymienione elementy instrukcji oddziela się od siebie odstępami. Oto przykład:

```
incr    i = i + 1    :(done)
...
done
```

Po zwiększeniu zmiennej **i** następuje przekazanie sterowania do instrukcji z etykietą **done**.

Niektóre działania w języku SNOBOL4 są **warunkowe** (ang. conditional); ich wykonywanie może kończyć się powodzeniem **(może się spełniać)** lub prowadzić do niepowodzenia. Powodzenie **(S)** — (ang. success) i niepowodzenie **(F)** — (ang. failure) mogą być wykorzystywane do sterowania programem za pomocą skoków warunkowych, jak niżej:

```
loop    EQ(i,j)     :S(done)
        i = i + 1
        obliczenia
        ...
                    :(loop)
done
```

Funkcja **EQ** jest nazywana **warunkiem** (ang. predicate). Dany warunek jest spełniony, jeśli **i** równa się **j**, w przeciwnym razie warunek **EQ** nie jest spełniony. Powyższa pętla kończy się przejściem do **done**, kiedy wartość **i** zrówna się z wartością **j**. Oznaczenia warunków numerycznych są takie same jak w Fortranie, przy czym w języku SNOBOL4 nie ogranicza się ich za pomocą kropek. Istnieją również warunki służące porównywaniu łańcuchów i innych typów danych.

W instrukcji można przewidzieć **skok** zarówno w wypadku powodzenia, jak i niepowodzenia, na przykład:

```
IDENT(si,s2)        :S(yes)F(no)
```

Warunek **IDENT** spełnia się, jeśli łańcuchy **s1** i **s2** są identyczne — w przeciwnym razie występuje niepowodzenie.

W języku SNOBOL4 struktury sterowania jako wydzielone konstrukcje nie występują; muszą być symulowane przy użyciu operacji warunkowych i skoków. Na przykład, pętla służąca organizacji obliczeń dla wszystkich wartości poczynając od **I**, a na pewnej wartości **limit** kończąc, może zostać zaprogramowana następująco:

```
        const = 1
test    GT(count,limit) :S(complete)
        ...
        count = count + 1 :(test)
complete
```

Wykonanie programu w języku SNOBOL4 rozpoczyna się od jego pierwszej instrukcji. Koniec programu stanowi instrukcja składająca się z wyróżnionej etykiety **END**. Program przestaje działać z chwilą osiągnięcia instrukcji **END**.

Dopasowywanie wzorców

SNOBOL4 stosuje dopasowywanie wzorców do analizy i dekompozycji łańcuchów. **Wzorzec** w języku SNOBOL4 jest kategorią danych charakteryzującą klasę łańcuchów, które można do niego „dopasować". Instrukcja dopasowywania ma postać:

```
s ? p
```

w której **s** jest **podmiotem**, a **p** oznacza **wzorzec**. W standardowym języku SNOBOL4 działanie dopasowywania wzorca nie jest oznaczane jawnie operatorem, podobnie jak spinanie, zatem znak **?** nie występuje. Jednak wiele dialektów języka SNOBOL4 posługuje się jawnym operatorem **?**, co jest czynione i tutaj, dla większej jasności.

W najprostszym wypadku wzorzec może być po prostu łańcuchem, tak jak poniżej:

```
line ? "comment"
```

gdzie bada się czy **line** zawiera podłańcuch **"comment"**.

Instrukcja dopasowywania wzorca jest **warunkowa**; jej powodzenie — bądź nie — zależy od dopasowania (bądź niedopasowania) wzorca. Może to być wykorzystanie w warunkowym sterowaniu, jak na przykład:

```
line ? "comment"        :S(skip)
```

Dopasowany do wzorca podłańcuch można zastąpić innym, powodując zmianę wartości podmiotu. Dzieje się tak w instrukcji zastąpienia, będącej połączeniem dopasowywania wzorca i przypisania:

```
s1 ? p = s2
```

Przykładowo,

```
line ? "if" = " "
```

powoduje usunięcie z podmiotu **line** pierwszego (od lewej) łańcucha **"if"**, wskutek zastąpienia go łańcuchem pustym.

Podobnie:

```
del     line ? "if" = ""    :S(del)
```

usuwa wszystkie wystąpienia łańcucha **"if"** w **line**. Zauważmy, że powyższy postulat nie jest spełniony i wykonywanie programu jest podejmowane od następnej instrukcji, gdy **line** nie zawiera już ani jednego **"if"**.

Prócz łańcuchów, istnieje wiele wzorców wbudowanych w język SNOBOL4 na stałe, między inymi:

ARB — dopasowuje dowolny łańcuch,
BAL — dopasowuje łańcuch zrównoważony co do liczby nawiastów,
LEN(i) — dopasowuje **i** znaków,
TAB(i) — dopasowuje tekst do **i**-tego znaku,
REM — dopasowuje resztę znaków podmiotu,
ANY(s) — dopasowuje dowolny spośród znaków w **s**,
BREAK(s) — dopasowuje część podmiotu aż do wystąpienia znaku z **s**,
SPAN(s) — dopasowuje ciąg znaków z **s**.
Na przykład:

```
line ? LEN(10)      :F(short)
```

jest spełnione tylko wówczas, gdy **line** zawiera przynajmniej 10 znaków, a warunek:

```
line ? ANY("aeiou")     :S(vowel)
```

jest spełniony jedynie wówczas, gdy **line** zawiera którąś z wymienionych w łańcuchu małych liter (samogłosek).

Dopasowywanie może zachodzić kolejno dla kilku wzorców. W tym celu zapisuje się je jeden za drugim, tak jak przy spinaniu łańcuchów. Z chwilą dopasowania któregoś wzorca, następuje przemieszczenie punktu odniesienia w podmiocie poza dopasowany podłańcuch. Na przykład:

```
line ? ("x" LEN(1) "x")     :S(xx)
```

spełni się tylko wówczas, gdy **line** zawiera znaki **"x"** podzielone dowolnym innym — lecz jednym — znakiem.

Dopasowanie można zadawać alternatywnie, oddzielając wzorce za pomocą pionowych kresek, jak niżej:

```
text ? ("teal" | "coot")    :S(bird)
```

co jest spełnione, gdy **text** zawiera podłańcuch **„teal"** albo **"coot"**.

Wzorce można określać w sposób dowolnie złożony za pomocą kombinowania wzorców elementarnych (wbudowanych) w alternatywach i operacjach spinania, jak w przykładowym wzorcu:

```
(("teal" | "coot") ARB "home")
```

który daje się dopasować do napisów **„teal"** lub **"coot"**, po których występuje łańcuch **„home"**, być może oddzielony od nich pewną liczbą innych znaków (dopasowanych przez **ARB**).

Dążąc do dopasowania podmiotu, algorytm dopasowywania wzorców języka SNOBOL4 wykorzystuje wszystkie określone we wzorcu możliwości na wszystkie możliwe sposoby. Jeśli pewien element wzorca daje się dopasować, lecz następne wymaganie nie jest spełnione, to w algorytmie dopasowywania następuje nawrót i próba dopasowania czegoś innego do poprzedniego elementu, jeśli taka możliwość istnieje.

Aby ustalić, które podłańcuchy są aktualnie dopasowywane, z elementem wzorca można stowarzyszyć zmienną, pisząc:

```
p . v
```

Działanie wzorca zawierającego takie wyrażenie polega na tym, że podłańcuch dopasowany do **p** zostaje przypisany **v**.

Oto przykład:

```
line ? (LEN(10) . f10)      :F(none)
```

w którym następuje przypisanie zmiennej **f10** pierwszych 10 znaków podmiotu **line**. Jeśli **line** ma mniej niż 10 znaków, to próba dopasowania kończy się niepowodzeniem i wartość **f10** pozostaje nie zmieniona. Kropkę można uznać w ten sposób za operator „warunkowego" przypisania.

Wzorce stanowią kategorię danych, które mogą podlegać swoistemu obliczaniu, przypisywaniu zmiennym i — przez odwołania do tych zmiennych — wtórnemu użyciu w wyrażeniach dopasowywania innych wzorców. Przykład:

```
bird = ("teal" : "coot") ARB "home"
...
text ? bird         :S(pond)
```

Tym samym wzorce mogą być konstruowane „na boku" i używane w dowolnym miejscu w programie na podobieństwo procedur.

Wejście-wyjście

Wprowadzanie i wyprowadzanie danych w języku SNOBOL4 odbywa się niejawnie. Nazwa **OUTPUT** jest stowarzyszona ze standardowym plikiem wyjściowym. Przypisanie wartości nazwie **OUTPUT** jest równoznaczne z wypisaniem tej wartości na wyjściu.

Nazwa **INPUT** symbolizuje standardowy plik wejściowy. Każdorazowe pojawienie się nazwy **INPUT** w kontekście wymagającym przekazania wartości powoduje użycie w obliczeniach treści nowego wiersza wprowadzonego z wejścia.

Z chwilą osiągnięcia końca pliku wejściowego, próba użycia wartości **INPUT** prowadzi do niespełnienia wyrażenia, w którego kontekście była podjęta. Na przykład, aby przekopiować plik z wejścia na wyjście wystarczy użyć jednej instrukcji:

```
copy    OUTPUT = INPUT      :S(copy)
```

Jest możliwe dokonywanie powiązań na wejściu i wyjściu między innymi zmiennymi i plikami. Zmienne reprezentujące pliki we-wy mogą występować w dowolnych miejscach w wyrażeniach. Przykładowo:

```
INPUT ? sentence      :S(okay)
```

powoduje przeczytanie następnego wiersza z wejścia i przejście do **okay**, jeśli wzorzec **sentence** uda się do nowo przeczytanego wiersza dopasować. Podobnie:

```
s ? ANY(INPUT)      :S(check)
```

powoduje przejście do **check**, gdy **s** zawiera bodaj jeden ze znaków występujących w kolejnym wierszu wejściowym, a

```
s ? (p . OUTPUT)
```

wypisuje na wyjściu podłańcuch **s** dopasowany do wzorca **p** przy założeniu, że dopasowanie się powiodło.

Struktury

Wiele problemów przetwarzania napisów wymaga posługiwania się strukturami danych. SNOBOL4 ma trzy rodzaje struktur stanowiących agregaty zmiennych (elementów): **definiowane reprezentacje danych, tablice** (ang. array) i **tabele** (ang. table). Reprezentacje definiowane (rekordy) są osiągalne za pomocą ponazywanych pól, dostęp do tablic odbywa się przez wskazanie pozycji elementu (dostęp swobodny), w tabelach o wyborze decydują wartości-etykiety elementów (dostęp skojarzeniowy). Struktury mogą być **niejednorodne** (ang. nonhomogeneous), tzn. typy wartości poszczególnych elementów struktur mogą być różne. Struktur nie deklaruje się, są one utworzone w trakcie działania programu. Rozmiary struktur są określane w chwili ich tworzenia, wtedy również następuje automatyczne przydzielenie niezbędnej dla nich pamięci.

Funkcja **DATA(s)** tworzy **zdefiniowany typ danych (typ rekordowy)** według informacji zawartych w prototypie **s**. Wartość **s** określa nazwę typu i nazwy przynależnych mu pól. Na przykład:

```
DATA("word(value, type, count)")
```

tworzy nowy typ **word** z trzema polami: **value, type** i **count**. Każde pole jest zmienną. Tak więc, **word** jest agregatem (rekordem) trzech zmiennych. Po wykonaniu funkcji **DATA** tworzenie obiektów nowo zdefiniowanego typu odbywa się za pomocą wywołań funkcji, zgodnych z określonym prototypem. W ten sposób:

```
x = word("lamp", "noun", 0)
```

powoduje przypisanie zmiennej **x** obiektu typu **word**, w którym pole **value** ma wartość **"lamp"**, pole **type"** przyjmuje wartość **"noun"**, a pole **count** równa się **0**.

Odniesienia do pól są realizowane za pomocą funkcji odpowiadającym nazwom pól. Na przykład:

```
OUTPUT = type(x)
```

przesyła na wyjście napis **noun**, a

```
count(x) = count(x) + 1
```

dodaje 1 do pola **count** w (rekordzie) **x**.

Funkcja **ARRAY(s)** tworzy tablicę, której rozmiar i kształt są zadatkowane przez wartość **s**. Na przykład:

```
a = ARRAY(100)
```

tworzy tablicę stuelementową, a

```
list = ARRAY(INPUT)
```

określa tablicę o wymiarach przeczytanych z pliku wejściowego.

Elementy tablicy są uzyskiwane (indeksowane) według ich miejsca. Na przykład:

```
a[30] = INPUT      :F(cof)
```

powoduje przypisanie następnego wiersza z pliku wejściowego trzydziestemu elementowi tablicy **a**.

Element tablicy może mieć wartość dowolną — może nią być sama tablica, jak w przykładzie:

```
a[1] = a
```

Jest to możliwe dzięki temu, że tablice są reprezentowane za pomocą wskaźników do aktualnej struktury danych.

Wskaźnik przekraczający granice wymiaru tablicy powoduje niepowodzenie obliczeń; na przykład:

```
        i = 1
write   OUTPUT = a[i]      :F(done)
        i = i + 1
done
```

spowoduje wyprowadzenie na wyjście wszystkich wartości tablicy **a**. Zauważmy, że obywamy się tutaj bez wiedzy o liczbie elementów tablicy **a**.

Tablice mogą mieć wiele wymiarów. Przedziały wskaźników można określać dla każdego wymiaru oddzielnie. Na przykład:

```
box = ARRAY("—4:4,—7:0,10:20")
```

powoduje utworzenie trójwymiarowej tablicy, w której pierwszy wymiar przebiega przedział od —4 do 4, drugi od —7 do 0, a trzeci — od 10 do 20. Odwołania do tablic wielowymiarowych są wykonywane przez podanie wykazu wskaźników, jak w przykładzie:

```
OUTPUT = box[0,—3,15]
```

Tabela stanowi zespół zmiennych identyfikowanych nie przez pozycję, lecz za pomocą wartości (etykiety elementu). Utworzenie tabeli przypomina zainicjowanie tablicy:

```
words = TABLE()
```

W chwili utworzenia tabela jest pusta — nie zawiera żadnych elementów.

Odwołanie do elementu tabeli jest podobne do odwołania do elementu tablicy, przy czym wskaźnik nie musi być liczbą całkowitą. Przykładowo:

```
words["the"] = "article"
```

powoduje przypisanie wartości **"article"** elementowi **"the"** w tabeli **words**. Podobnie,

```
OUTPUT = words["the"]
```

wypisuje bieżącą wartość elementu **"the"** z tabeli **words**.

Każdorazowo przy użyciu nowego wskaźnika tabeli, jej rozmiar zwiększa się automatycznie. Nie istnieje wewnętrzne ograniczenie rozmiaru tabeli.

Tabela może zostać przekształcona na tablicę za pomocą przypisania:

```
a = SORT(t)
```

Funkcja **SORT** zamienia tabelę na dwuwymiarową tablicę, w której pierwsza kolumna zawiera uporządkowane wartości wejściowe (wskaźniki) tabeli **t**, a druga — odpowiadające im wartości elementów tabeli.

Przykład

Użycie tabel i tablic w połączeniu z przetwarzaniem łańcuchów zilustrowano w programie sporządzającym zestawienie częstości występowania poszczególnych słów w tekście (wydruk). Program składa się z dwu części. Część pierwsza czyta plik, wyodrębnia słowa i zlicza je w tabeli. Część druga przekształca tabelę słów na tablicę i wyprowadza wyniki.

Załóżmy na użytek przykładu, że „słowo" oznacza łańcuch złożony wyłącznie z wielkich i małych liter:

```
letters = "ABC ... XYZabc ... xyz"
find = BREAK (letters) (SPAN(letters) . word)
```

```
        letters = "ABC...XYZabc...xyz"
        find = BREAK(letters) (SPAN(letters) . word)
        count = TABLE()
nextl   line = INPUT                    :F(part2)
nextw   line ? find = ""                :F(nextl)
        count[word] = count[word] + 1
+                                       :(nextw)
part2   list = SORT(wcount)
        i = 1
nextr   OUTPUT = list[i,1] " : " list[i,2]
+                                       :F(END)
        i = i + 1                       :(nextr)
END
```

Wejście	Wyjście
Water, water, every where,	And : 1
And all the boards did shrink;	Nor : 1
Water, water, every where,	Water : 2
Nor any drop to drink.	all : 1
	any : 1
	boards : 1
	did : 1
	drink : 1
	drop : 1
	every : 2
	shrink : 1
	the : 1
	to : 1
	water : 2
	where : 2

Program zliczania słów w języku SNOBOL4

Wzorzec **find** dopasowuje znaki do wystąpienia pierwszej litery, a następnie ciąg kolejnych liter, przypisując go zmiennej **word**.

Pierwsza część programu ma postać:

```
        count = TABLE()
nextl   line = INPUT        :F(part2)
nextw
        line ? find = ""    :F(nextl)
        count [word] = count [word] + 1
+                           :(nextw)
```

Instrukcja opatrzona etykietą **nextw** jest powtarzana dla każdego wiersza wejściowego, powodując usuwanie znaków do chwili, gdy w przeczytanym wierszu nie pozostanie ani jeden znak. Wskutek użycia przy dopasowywaniu wzorca **find** (patrz wyżej), każde dopasowane słowo zostanie zapamiętane pod zmienną **word**. W następnym wierszu programu wykorzystuje się to słowo do indeksowania tabeli, której wartości są licznikami wystąpień poszczególnych słów. Znak „+" w pierwszej kolumnie wskazuje na kontynuację długiego wiersza. Po osiągnięciu końca pliku wejściowego następuje przejście do **part2**, gdzie wyprowadzenie wyników przebiega następująco:

```
part2   list = SORT(count)
        i = 1
nextr   OUTPUT = list[i,1] ":" list[i,2]
+                           :F(END)
        i = i + 1           :(nextr)
```

Na początku tabela jest zamieniana na tablicę. Pierwsza kolumna otrzymanej tablicy zawiera słowa wyodrębnione na wejściu, a druga zawiera odpowiednie liczniki. Zauważmy, że nie ma potrzeby wnikać w liczbę ani rodzaje słów na wejściu. Program działa dla każdego pliku.

Cechy ezoteryczne

SNOBOL4 ma pewne właściwości nie spotykane w innych językach programowania. Jest możliwe przetwarzanie samych nazw zmiennych (odwołania pośrednie) oraz tworzenie ich w czasie wykonywania programu. Są dozwolone wtórne definicje funkcji i operatorów wbudowanych. Kompilacja i modyfikacje dostępne w toku wykonywania programu pozwalają na przeobrażanie się programu w czasie jego wykonania.

Operator $s pośredniego odwoływania się do zmiennej wykorzystuje **wartość** s jako nazwę. Na przykład:

$"sum" = 0

jest równoważne przypisaniu:
sum = 0

Odwołania pośrednie mogą być użyte do identyfikacji przez kojarzenie, jak w instrukcji:

$color = $color + 1

która zwiększa wartość **blue**, jeśli wartość **color** wynosi **"blue"**, wartość **red**, jeśli wartość **color** równa się **„red"** itd. Odnotujemy podobieństwo odwołań pośrednich do wykorzystania łańcuchów w roli wskaźników w tabelach. Dla przykładu, zamiast uprzednio zademontrowanego mechanizmu odwołań pośrednich mogłaby być użyta konstrukcja:

```
count = TABLE()
...
count[color] = count[color] + 1
```

Funkcja **CODE(s)** kompiluje łańcuch s, a jej wynik staje się częścią wykorzystywanego programu. Na przykład:

CODE("err OUTPUT = errval : (END)")
powoduje dołączenie instrukcji:

err OUTPUT = errval :(END)

do programu. Jeśli w programie istniała już instrukcja z etykietą **err**, to nowo skompilowana instrukcja zastąpi ją.

* * *

Oryginalna implementacja języka SNOBOL4 została ukończona w Bell Laboratories w 1968 roku i jest znana pod nazwą **SIL**, co jest skrótem słów: **S**NOBOL4 **I**mplementation **L**anguage. Jest to specjalizowany język zaprojektowany wyłącznie do celów realizacji języka SNOBOL4. Głównymi cechami języka SIL są: przenośność i elastyczność, choć jego implementacja nie odznacza się zbytnią efektywnością. Realizacja SIL zawiera wszystkie właściwości języka SNOBOL4. Wdrożono ją na wielu komputerach, poczynając od wielkich konfiguracji, a na minikomputerach kończąc. Niedawno udostępniono dwie implementacje pracujące pod kontrolą systemu operacyjnego MS-DOS na komputerach osobistych. W chwili obecnej (1986 r.) w stadium opracowywania znajduje się implementacja języka SNOBOL4 dla komputerów Apple Macintosh i innych wyposażonych w procesory MC68000. Implementacja SIL jest dostępna powszechnie, bez opłat.

Istnieją również inne realizacje języka SNOBOL4. Jedną z najpowszechniej używanych jest **MACRO SPITBOL**, stanowiący zarówno dialekt SNOBOL4, jak i system implementowania. MACRO SPITBOL ma większość cech standardowego języka SNOBOL4, włączając funkcję CODE, ale nie zezwala na redefiniowanie funkcji i operatorów wbudowanych. Prócz tego MACRO SPITBOL wprowadza kilka rozszerzeń w stosunku do języka standardowego. Realizacja MACRO SPITBOL jest ukierunkowana na wysoką wydajność i przenośność — wytwarza zwięzłe programy (przekłady) o dużej szybkości działania. Jest godna polecenia przy produkcji oprogramowania, daje szczególnie dobre wyniki na komputerach o małych pamięciach. Również i ona została wdrożona na licznych komputerach od wielkich po personalne, nadzorowane przez MS-DOS.

Tłumaczył i opracował:

ZDZISŁAW PŁOSKI

LITERATURA

[1] Emmer M. B.: SNOBOL4 — The SNOBOL4 Language for the Personal Computer User. Prentice-Hall, Englewood Cliffs (NJ), 1985

[2] Gimpel J. F.: Algorithms in SNOBOL4. Wiley and Sons, New York, 1976

[3] Griswold A. E.: String and List Processing in SNOBOL4 — Techniques and Applications. Prentice-Hall, Englewood Cliffs (NJ), 1973

[4] Griswold R. E., Poage J. F., Polonsky I. P.: The SNOBOL4 Programming Language. Prentice-Hall, Englewood Cliffs (NJ), 1971.

MACIEJ M. SYSŁO
Instytut Informatyki
Uniwersytet Wrocławski

POLSKIE TOWARZYSTWO INFORMATYCZNE

Algorytmy kombinatoryczne i ich efektywność (3)

Kolorowanie wierzchołków grafu

W tej części artykułu omówiono kolorowanie wierzchołków grafu, które sąsiednim wierzchołkom przyporządkowuje różne kolory. Formalnie, **k-kolorowaniem** grafu $G = (V,E)$ nazywa się funkcję $f: V \xrightarrow{na} \{1, 2, \ldots, k\}$, spełniającą $f(i) \# f(j)$, jeśli $\{i,j\} \in E$. **Liczbą chromatyczną** $\chi(G)$ grafu G nazywa się najmniejsze k, dla którego G ma k-pokolorowanie. Każdy graf można pomalować $n = |V|$ kolorami. Wynika stąd, że do wyznaczenia liczby $\chi(G)$ wystarczy rozpatrzyć n^n funkcji kolorujących wierzchołki grafu. To oszacowanie liczby możliwych pokolorowań może być zmniejszone do n! przez wyeliminowanie tych, które powstają z innych przez prostą zmianę nazw kolorów (por. [5], str. 110—112).

PROBLEM k-POKOLOROWANIA GRAFU

Dane:

Graf $G = (V,E)$ i liczba naturalna $k (1 \leqslant k \leqslant |V|)$.

Pytanie:

Czy G może być pomalowany k kolorami?

Zauważmy, że potrafiąc efektywnie rozwiązywać powyższy problem dla dowolnego k, można znaleźć liczbę chromatyczną grafu. W tym celu wystarczy rozpatrzyć k-pokolorowanie dla kolejnych $k = n, n-1, n-2, \ldots$, aż do otrzymania negatywnej odpowiedzi. Ostatnie k, dla którego G może być pomalowany k kolorami, jest liczbą chromatyczną grafu G. Przez wzgląd na zastosowania, interesujące są metody rozwiązywania problemu k-pokolorowania, które w wypadku pozytywnej odpowiedzi wyznaczają także jedno z k-pokolorowań.

Wśród najpopularniejszych zastosowań problemu kolorowania grafu, można wymienić układanie rozkładów zajęć oraz harmonogramu prac [3, 8]. Jednak swoją największą popularność kolorowanie grafu zawdzięcza problemowi czterech kolorów, który polega na określeniu najmniejszej liczby barw potrzebnych do pomalowania administracyjnej mapy państw tak, aby żadne dwa państwa sąsiadujące ze sobą granicą dłuższą od punktu nie były tych samych kolorów. Mniej więcej od połowy zeszłego wieku, aż do 1976 roku znane było przypuszczenie, że każda mapa może być pomalowana czterema kolorami i dopiero w połowie 1976 roku K. Appel i W. Haken, korzystając z pomocy komputera, wykazali prawdziwość tej hipotezy. Malowanie mapy jest w gruncie rzeczy kolorowaniem specjalnych grafów. Każdej mapie państw można bowiem przyporządkować graf, którego wierzchołki odpowiadają państwom (są na przykład ich stolicami), a dwa wierzchołki sąsiednie, jeśli odpowiadające im państwa graniczą ze sobą. W ten sposób malowanie mapy sprowadza się do malowania grafów sąsiedztwa państw [1, 8, 9]. Są to tzw. grafy planarne.

Kolorowanie grafów należy do najtrudniejszych problemów kombinatorycznych. Ponieważ algorytmy dokładne są mało efektywne, wiele uwagi poświęca się efektywnym algorytmom heurystycznym. Jednym z niewielu rezultatów o dokładnym kolorowaniu jest twierdzenie Königa charakteryzujące grafy, które mogą być malowane dwoma kolorami:

Graf G jest dwuchromatyczny wtedy i tylko wtedy, gdy zawiera co najmniej jedną krawędź i nie zawiera cykli nieparzystej długości.

Twierdzenie to można udowodnić podając algorytm, który maluje dwoma kolorami każdy graf nie zawierający nieparzystych cykli (cykl nieparzystej długości nie może być pomalowany dwoma kolorami). Zatem charakteryzacja dwuchromatycznych grafów jest efektywna. Nie jest natomiast znana żadna efektywna charakteryzacja grafów k-chromatycznych dla $k \geqslant 3$, nawet w klasie grafów planarnych, tzn. odpowiadającym mapom.

Jeśli graf G zawiera **klikę** (tj. podgraf pełny) złożoną z k wierzchołków, to $\chi(G) \geqslant k$, gdyż każdy wierzchołek tej kliki musi być pomalowany innym kolorem. Zatem:

(1) $\omega(G) \leqslant \chi(G)$,

gdzie $\omega(G)$ jest **liczbą klikową** grafu, tj. mocą największej kliki grafu G. Z drugiej zaś strony:

(2) $\chi(G) \leqslant \Delta(G) + 1$,

gdzie $\Delta(G)$ jest maksymalnym stopniem wierzchołka w grafie G. Nierówność ta wynika ze spostrzeżenia, że spośród $k = \Delta(G)+1$ dostępnych kolorów w każdym wierzchołku v, zawsze jeden kolor jest wolny, gdyż v ma mniej niż k sąsiadów. Wspólną cechą obu oszacowań jest istnienie wielu nieskończonych klas grafów, dla których stają się one dowolnie złe. J. Mycielski i W. T. Tutte skonstruowali nieskończony ciąg grafów F_n, które nie zawierają trójkątów, czyli $\omega(F_n) = 2$ i $\chi(F_n) = n$ (na rys. 1 przedstawiono graf F_4). Z drugiej strony, gwiazda S_n z n promieniami jest grafem dwuchromatycznym i $\Delta(S_n) = n$. Oszacowania (1) i (2) należą jednak do różnych klas. Wyznaczanie liczby $\omega(G)$ jest bowiem tak samo trudne jak obliczanie $\chi(G)$. Natomiast wartość $\Delta(G)$ może być znaleziona w czasie $0(m)$.

SEKWENCYJNE METODY KOLOROWANIA

Algorytm sekwencyjny rozpatruje wierzchołki grafu w ustalonej kolejności i maluje kolejny wierzchołek kolorem o możliwie najmniejszym numerze. Zatem kolory są przydzielane w możliwie najoszczędniejszy sposób. Zauważmy, że tak określona ogólna idea postępowania nie wyklucza możliwości otrzymywania optymalnych pokolorowań. Wystarczy w tym celu przyjąć, że wierzchołki grafu są uporządkowane zgodnie z kolorami: najpierw wierzchołki koloru 1, potem koloru 2 itd.

ALGORYTM SEKWENCYJNY

Dane:

Graf G oraz uporządkowanie jego wierzchołków $v_1, v_2, \ldots, \ldots, v_n$

Wynik:

Przyporządkowanie f określające pokolorowanie grafu G otrzymane metodą sekwencyjną.

begin

$f(v_1) \leftarrow 1$;

for $i \leftarrow 2,3,\ldots,n$ **do**

$f(v_i) \leftarrow \{\min k : k \geqslant 1$ **oraz** $f(v_j) \neq k$ **dla każdego** $v_j (1 \leqslant j < i)$ **sąsiedniego z** $v_i\}$

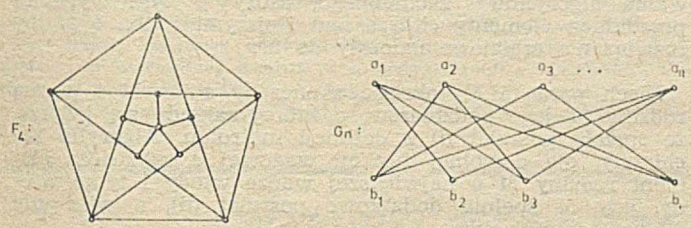
end

Liczbę kolorów użytych przez ten algorytm można oszacować następująco. Dla każdego i prawdziwa jest nie-

równość $f(v_i) \leqslant i$, gdyż i-ty w kolejności wierzchołek może być zawsze pomalowany i-tym kolorem. Z drugiej zaś strony, kolorami zabronionymi dla $f(v_i)$ są kolory sąsiadów wierzchołka v_i, wśród wierzchołków $v_1, v_2, \ldots, v_{i-1}$, których jest co najwyżej $\deg(v_i)$, gdzie $\deg(v)$ jest liczbą wierzchołków sąsiednich z v. Zatem $f(v_i) \leqslant \deg(v_i)+1$. Otrzymuje się więc $f(v_i) \leqslant i$ oraz $f(v_i) \leqslant \deg(v_i)+1$, czyli $f(v_i) \leqslant \min\{i, \deg(v_i)+1\}$, dla $i = 1, 2, \ldots, n$. Niech $h(G)$ oznaczá liczbę kolorów użytych przez algorytm sekwencyjny zastosowany do uporządkowania $v_1, v_2, \ldots, v_n$. Wówczas

$$\chi(G) \leqslant h(G) \leqslant \max_{1 \leqslant i \leqslant n} \min\{i, \deg(v_i)+1\} = h_s(G)$$

Wielkość $h_s(G)$ jest jedynie oszacowaniem liczby $h(G)$. W praktyce, co wykazały wielokrotnie wykonane eksperymenty, liczba użytych kolorów $h(G)$ jest znacznie mniejsza niż $h_s(G)$. Oszacowanie $h_s(G)$ dla uporządkowania wierzchołków spełniającego $\deg(v_1) \leqslant \deg(v_2) \leqslant \ldots \leqslant \deg(v_n)$ staje się znanym już oszacowaniem $h_s(G) = \Delta(G)+1$. Postać wyrażenia określającego $h_s(G)$ sugeruje natomiast, że $h_s(G)$ jest najmniejsze dla odwrotnego uporządkowania wierzchołków. Metodę sekwencyjną z uporządkowaniem wierzchołków spełniającym nierówności $\deg(v_1) \geqslant \deg(v_2) \ldots \geqslant \deg(v_n)$ zaproponowali jako pierwsi D. J. A. Welsh i M. B. Powell. Niech $h_{WP}(G)$ oznacza wartość $h_s(G)$ w tym wypadku. Istnieją niestety klasy grafów, dla których $h_{WP}(G)$ może dowolnie dużo różnić się od $\chi(G)$. Dla przykładu można zastosować algorytm sekwencyjny do pomalowania wierzchołków grafu G_n z rys. 1 w kolejności $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$, będącej uporządkowaniem Welsha-Powella. Każdy graf G_n jest dwuchromatyczny, gdyż $f(a_j) = 1$ i $f(b_j) = 2$ jest pokolorowaniem właściwym. Natomiast algorytm sekwencyjny używa aż n kolorów, a także $h_{WP}(G_n) = n$.



Rys. 1. Przykład kolorowanego grafu

Wyprowadzając oszacowanie $h_s(G)$ wykazano, że $f(v_i) \leqslant \deg(v_i)+1$. W rzeczywistości prawdziwa jest nierówność $f(v_i) \leqslant \deg_i(v_i)+1$, gdzie $\deg_i(v_i)$ jest stopniem wierzchołka v_i w podgrafie generowanym przez wierzchołki $\{v_1, v_2, \ldots, v_i\}$. Otrzymuje się stąd jeszcze jedno oszacowanie liczby kolorów użytych przez algorytm sekwencyjny:

$$\chi(G) \leqslant h(G) \leqslant h'_s(G) = \max_{1 \leqslant i \leqslant n} \deg_i(v_i)+1.$$

D. Matula udowodnił, że kolejność wierzchołków minimalizująca wielkość $h'_s(G)$ po wszystkich n! możliwych uporządkowaniach może być wyznaczona w następujący prosty sposób:

(i) za v_n przyjmij wierzchołek o minimalnym stopniu w G;
(ii) dla $i = n-1, n-2, \ldots, 1$ wybierz wierzchołek v_i o minimalnym stopniu w podgrafie indukowanym przez $V_i = V - \{v_n, v_{n-1}, \ldots, v_{i+1}\}$.

Algorytm sekwencyjny zastosowany do powyższego uporządkowania wierzchołków jest nazywany algorytmem Matuli. Podobnie jak w algorytmie Welsha-Powella, także i w tym wypadku liczba użytych kolorów jest znacznie mniejsza niż wartość oszacowania $h'_s(G)$; istnieją też klasy grafów o ograniczonej liczbie chromatycznej, które tą metodą są malowane bardzo źle. Dla uporządkowania Matuli prawdziwa jest równość: $\deg_i(v_i) = \min_{1 \leqslant j \leqslant i} \deg_i(v_j)$

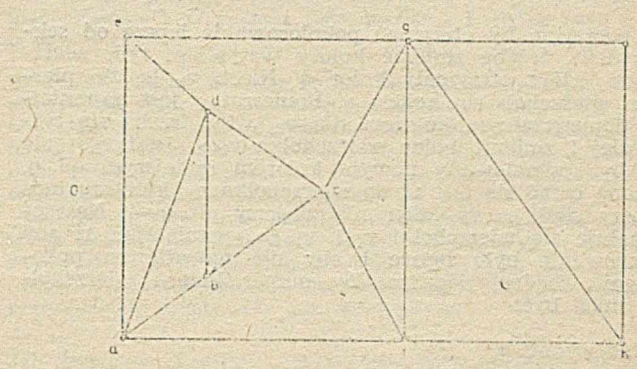
Zatem

$$\chi(G) \leqslant h'_s(G) = 1 + \max_{1 \leqslant i \leqslant n} \min_{v_j \in V_i} \deg_i(v_j)$$

gdzie $V_n = V$. Stąd otrzymuje się oszacowanie udowodnione po raz pierwszy przez Szekeresa i Wilfa:

$$\chi(G) \leqslant 1 + \max_{H \subseteq G} \delta(H)$$

gdzie maximum jest po wszystkich podgrafach H grafu G, a $\delta(H) = \min_{v \in V(H)} \deg(v)$. Ostatnie oszacowanie jest bardzo użyteczne w wypadku grafów, w których każdy podgraf ma wierzchołek o niewielkim stopniu. Na przykład, dla grafu planarnego G jest $\delta(G) \leqslant 5$.



Rys. 2. Drugi przykład kolorowanego grafu

Wyniki zastosowania algorytmów Welsha-Powella i Matuli do pomalowania grafu G z rys. 2 przedstawiono w tabeli. W tym wypadku $\chi(G) = 3$, gdyż G zawiera trójkąt, a więc $\omega(G) \geqslant 3$; zatem pokolorowanie otrzymane metodą Matuli jest optymalne. Pokolorowanie optymalne otrzymuje się także dla leksykograficznego uporządkowania wierzchołków, a więc dla kolejności a, b, c, d, e, f, g, h, i, która nie jest ani uporządkowaniem Welsha-Powella, ani uporządkowaniem Matuli.

Wyniki zastosowania algorytmów Welsha-Powella i Matuli do pomalowania grafu z rys. 2

Algorytm											Liczba użytych kolorów	Oszacowanie
Welsha-Powella	Kolejność wierzchołków	g	a	c	d	f	b	e	h	i	4	$h_{WP}(G)=5$
	Kolory:	1	1	2	3	3	4	2	2	3		
Algorytm Matuli	Kolejność wierzchołków	a	b	d	c	e	f	g	h	i	Liczba użytych kolorów	Oszacowanie
	Kolory:	1	2	3	1	2	2	3	1	2	3	$h_M(G)=4$

Powyższy przykład potwierdza to, że wyznaczane pokolorowania są lepsze niż odpowiednie oszacowania wynikające z działania tych algorytmów. Czytelnik nie powinien mieć większego kłopotu z napisaniem efektywnych realizacji obu algorytmów sekwencyjnych. Oba algorytmy mogą być wykonane w czasie 0(m) [6, 8].

Algorytm sekwencyjny może być usprawniony przez wprowadzenie możliwości wymiany dwóch kolorów w pewnych podgrafach, dzięki czemu można uwolnić jeden z kolorów w sytuacji, gdy oryginalny algorytm jest zmuszony do użycia nowego koloru. Ta modyfikacja może być zrealizowana także w czasie wielomianowym, nie prowadzi jednak do polepszenia oszacowania liczby chromatycznej w ogólnym wypadku.

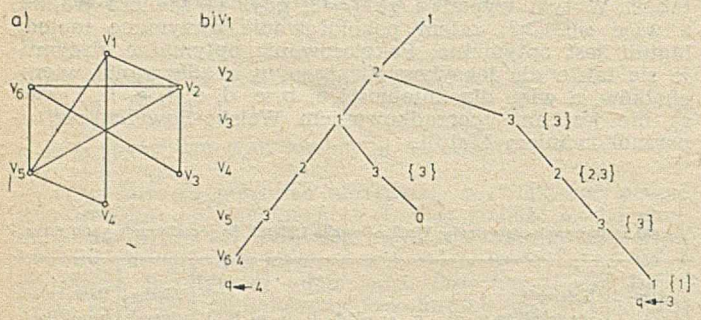
O słabości przybliżonych metod kolorowania grafów świadczyć może fakt udowodniony przez M. R. Garey'a i D. S. Johnsona [4]. Wykazali oni, że jeżeli będzie znana efektywna metoda kolorowania grafów G za pomocą mniej niż 2 $\chi(G)$ kolorów, to będzie można efektywnie wyznaczać chromatyczne pokolorowania grafów. Ponieważ przypuszcza się, że ta druga ewentualność jest mało prawdopodobna, należy uznać i tę pierwszą za prawie niemożliwą.

Algorytm sekwencyjny można łatwo rozszerzyć do metody znajdującej optymalne pokolorowania. Oto szkic jednego z takich algorytmów. W trakcie działania algorytmu wierzchołki grafu są rozpatrywane w tej samej kolejności $v_1, v_2, ..., v_n$, którą jednak można wybrać dowolnie. Za początkowe rozwiązanie przyjmuje się pokolorowanie otrzymane algorytmem sekwencyjnym. Niech q będzie liczbą użytych kolorów (dalej q będzie oznaczać liczbę kolorów użytych w najlepszym pokolorowaniu). Następne kroki algorytmu są realizacją przeglądania ciągu wierzchołków z nawracaniem (ang. backtracking), którego celem jest znalezienie możliwie najlepszego pokolorowania.

Na początku, by otrzymać pokolorowanie lepsze od sekwencyjnego, trzeba zmienić kolory wszystkim tym wierzchołkom, które otrzymały kolor q. Niech v_k będzie pierwszym wierzchołkiem koloru q. Ponieważ q jest najmniejszym dopuszczalnym kolorem dla v_k, więc trzeba się cofnąć dalej i zmienić kolor wierzchołka v_{k-1}. Jeśli v_{k-1} nie może być pomalowany nowym kolorem mniejszym od q, to trzeba cofać się tak aż do wierzchołka v_j, któremu można przydzielić nowy kolor nie większy niż q—1. Następnie maluje się wierzchołki $v_{j+1}, v_{j+2}, ...$ pamiętając, iż aby pokolorowanie było lepsze i aby nie natrafić na pokolorowanie zbędne, dopuszczalny numer koloru wierzchołka v_i musi być:

(i) mniejszy od q,
(ii) nie większy nik $\deg(v_i)+1$,
(iii) nie większy niż $l_{i-1}+1$,

gdzie l_{i-1} jest największym kolorem użytym do pomalowania wierzchołków $v_1, v_2, ..., v_{i-1}$.



Rys. 3. Drzewo przeglądania grafu dla algorytmu sekwencyjnego z optymalnym kolorowaniem

Warto prześledzić działanie tego algorytmu na przykładzie grafu z rys. 3a. Drzewo przeglądania przedstawiono na rys. 3b. Poziomy drzewa odpowiadają wierzchołkom, a cyfry w węzłach drzewa są numerami kolorów przydzielanych wierzchołkom grafu. Zbiory wyszczególnione przy węzłach drzewa zawierają numery kolorów dopuszczalnych dla poszczególnych wierzchołków grafu w chwili, gdy algorytm znajduje się w danym węźle drzewa.

Szczegółowy opis metody dokładnej oraz wielu jej modyfikacji można znaleźć w [8]. Zamieszczone tam wyniki eksperymentów przeprowadzonych na komputerze Amdahl 470 są raczej pesymistyczne. Nawet bardzo usprawniony algorytm dokładny nie był w stanie w rozsądnym czasie pomalować optymalnie niektórych grafów losowych o 45 wierzchołkach. Ten sam algorytm, zastosowany do grafu F_4 z rys. 1, znalazł rozwiązanie optymalne w pierwszej iteracji, a następnie spędził 4 ms (80 iteracji) na sprawdzeniu, że jest to rzeczywiście rozwiązanie optymalne.

* * *

W artykule starano się zilustrować zakres stosowania zachłannego sposobu rozwiązywania trzech wybranych problemów kombinatorycznych. Nieprecyzyjność i rozległość pojęcia zachłanności nie pozwala na formalizację obejmującą wszystkie wypadki, w których takie podejście jest stosowane. Ograniczmy więc uwagę do klasy problemów zawierającej problem najdłuższego drzewa rozpinającego (LST). Warto zauważyć, że w wypadku tych problemów każdy podzbiór rozwiązania dopuszczalnego jest także dopuszczalny: w problemie drzew dopuszczalny podzbiór krawędzi jest acykliczny, a w problemie plecakowym — dopuszczalny jest podzbiór towarów mieszczących się w plecaku. Różnica polega na tym, że tylko w pierwszym wypadku maksymalne (w sensie inkluzji) zbiory dopuszczalne są równoliczne, gdyż odpowiadają im drzewa rozpinające grafu. W problemie plecakowym, dopuszczalne zawartości plecaka mogą składać się z różnej liczby towarów. Te dwie właściwości dopuszczalnych rozwiązań problemu LST definiują bardzo ważną klasę obiektów kombinatorycznych, tzw. matroidów.

Matroidem nazywa się para $(M,\mathcal{M})$, w której M jest dowolnym zbiorem skończonym, a $\mathcal{M}$ jest rodziną jego niezależnych podzbiorów spełniających następujące warunki:
(i) zbiór pusty należy do $\mathcal{M}$ i jeżeli $I \in \mathcal{M}$ oraz $J \subseteq I$, to także $J \in M$;

(ii) dla każdego $N \subseteq M$, maksymalne podzbiory zbioru N należące do $\mathcal{M}$ są równoliczne.

Jeśli c jest wagową funkcją rzeczywistą określoną na M, to wagę podzbioru $I \subseteq \mathcal{M}$ określa się jako $c(I) = \sum_{e \in I} c(e)$.

Rozpatrując problem wyznaczenia najcięższego podzbioru w dowolnej rodzinie podzbiorów $\mathcal{M}$, poszukuje się zatem I^* takiego, że $c(I^*) = \max\{c(I): I \in \mathcal{M}\}$. Szczególnymi przypadkami tak ogólnie sformułowanego problemu są dwa pierwsze omówione problemy, dla których rodzina $\mathcal{M}$ składa się z podzbiorów dopuszczalnych, spełnia zatem warunek (i) z definicji matroidu. Następujące twierdzenie jednocześnie charakteryzuje matroidy i określa zakres stosowania algorytmu zachłannego, który w tym wypadku przedkłada elementy cięższe nad lżejszymi i dba o to, by rozwiązania częściowe należały do $\mathcal{M}$:

Niech para $(M,\mathcal{M})$ określa skończony zbiór M i pewną rodzinę $\mathcal{M}$ jego podzbiorów, o której zakłada się jedynie, że spełnia postulat (i) z definicji matroidu. Algorytm zachłanny, dla dowolnej funkcji wagowej c, wyznacza element rodziny $\mathcal{M}$ o największej wadze wtedy i tylko wtedy, gdy $\mathcal{M}$ spełnia dodatkowo postulat (ii), a więc gdy $(M,\mathcal{M})$ jest matroidem.

Twierdzenie to uzasadnia poprawność algorytmu zachłannego dla problemu LST oraz wyjaśnia, dlaczego prosty algorytm zachłanny nie zawsze dokładnie rozwiązuje problem plecakowy.

Czytelnika, zainteresowanego dalszymi właściwościami matroidów i algorytmu zachłannego, odsyłamy do elementarnego przedstawienia tych zagadnień w książkach [7] i [9].

LITERATURA

[1] Appel K., Haken W.: Zagadnienie czterech barw. Matematyką Współczesna, L. A. Steen red.), WNT, Warszawa, 1933

[2] Banachowski L., Kreczmar A.: Elementy analizy algorytmów. WNT, Warszawa, 1981

[3] Coffman E. G. (red.): Teoria szeregowania zadań. WNT, Warszawa, 1980

[4] Garey M. R., Johnson D. S.: Computers and Intractability. Freeman, San Francisco, 1979

[5] Jerszow A. P.: Wprowadzenie do teorii programowania. WNT, Warszawa, 1981

[6] Kubale M.: Analiza efektywności algorytmów kolorowania grafów. Matematyka Stosowana, Tom 19, str. 23—41, 1982

[7] Lipski W.: Kombinatoryka dla programistów. WNT, Warszawa, 1982

[8] Sysło M. M., Deo N., Kowalik J. S.: Discrete optimization algorithms with Pascal programs. Prentice-Hall, Englewood Cliffs (NJ), 1983

[9] Wilson R. J.: Wprowadzenie do teorii grafów. PWN, Warszawa, 1985.

WYDAWNICTWA NAUKOWO-TECHNICZNE

Andrews S. Tanenbaum „Sieci komputerowe". Warszawa 1988, wydanie I, 3800 egz., s. 580, cena 800 zł

Książka stanowi obszerną monografię wyczerpującą większość zagadnień dotyczących architektury i struktury współczesnych sieci komputerowych. Do opisu sieci — zarówno abstrakcyjnych, jak i już zrealizowanych — posłużono się tzw. modelem warstwowym. Omówiono zagadnienia sprzętowe, „czystego" przesyłania, sieci satelitarne, radiowe, lokalne i inne, a także zagadnienia niezawodności sieci i ich użytkowania, bezpieczeństwa transmisji, rozproszonych baz danych, szyfrowania i deszyfrowania wiadomości. Spis treści: Wprowadzenie; Topologia sieci; Warstwa fizyczna; Warstwa łącza danych; Warstwa sieciowa I (sieci typu punkt-punkt); Warstwa sieciowa II (sieci satelitarne i pakietowe, sieci radiowe); Warstwa sieciowa III (sieci lokalne); Warstwa transportowa i warstwa sesji; Warstwa prezentacji; Warstwa zastosowań.

Książka jest przeznaczona dla informatyków, inżynierów różnych specjalności interesujących się zagadnieniami teleinformatyki, pracowników nauki oraz studentów kierunków informatycznych.

Wacław Kasprzak, Bertold Lysik „Analiza wymiarowa — algorytmiczne procedury obsługi eksperymentu. Warszawa 1988, wydanie I, 800 egz. s. 244, cena 300 zł

W książce omówiono wybrane zagadnienia analizy wymiarowej i jej zastosowania w metrologii, teorii podobieństwa modelowego i w planowaniu eksperymentu. Podano algorytmiczne procedury projektowania eksperymentu i przetwarzania jego wyników, ilustrując teorię przykładami zastosowań w technice.

Książka zawiera również zagadnienia przydatności systemów komputerowych do obsługi eksperymentów oraz opisy oprogramowania.

Spis treści: Podstawy matematyczne badań empirycznych, przewodnik po literaturze; Analiza wymiarowa; Analiza wymiarowa a równania różniczkowe fizyki — funkcja wymiarowa złożona; Teoria podobieństwa modelowego; Modelowanie przybliżone, sprawdzanie kompletności zbioru niezmienników; Zmiana bazy wymiarowej — planowanie eksperymentu; Wybór postaci funkcji przydatnej do aproksymacji wyników eksperymentu; Identyfikacja modelu matematycznego z dokładnością do wymiarowych parametrów lub funkcji; Automatyzacja eksperymentu; Podstawy matematyczne procedur algorytmicznych biblioteki programów APTR.

Książka jest przeznaczona dla inżynierów pracujących w ośrodkach naukowo-badawczych i doktorantów. Mogą z niej korzystać specjaliści innych dziedzin zainteresowani metodologią eksperymentu oraz studenci.

Wolfgang Link „Jak mierzyć, sterować i regulować za pomocą Basicu?". Warszawa 1988, wydanie I, 19 700 egz., s. 156, cena 490 zł

W książce podano podstawowe informacje, zasady i przykłady z programami dotyczące wykorzystania mikrokomputerów do pomiarów różnych wielkości fizycznych, sterowania urządzeniami elektrycznymi oraz tworzenia automatycznych minisystemów regulacyjnych. Przedstawiono podstawowe dane najczęściej stosowanych interfejsów do operacji wejścia-wyjścia.

Spis treści: Interfejsy; Pomiary, sterowanie, regulacja, automatyczne pomiary za pomocą komputera wyposażonego w Basic; Zastępowanie sprzętu oprogramowaniem; Testowanie; Zakłócenia.

Książka jest przeznaczona dla szerokiego kręgu czytelników — użytkowników popularnych mikrokomputerów.

Jan Bielecki „Fortran". Warszawa 1988, wydanie I, 14 700 egz., s. 204, cena 680 zł

Książka zawiera opis standardu języka Fortran 77 — nowoczesnej wersji popularnego języka programowania Fortran. Szczegółowo omówiono wszystkie konstrukcje języka, a tekst wzbogacono przykładami.

Spis treści: Elementy języka; Specyfikacje; Wyrażenia i przypisania; Instrukcje sterujące; Instrukcja wejścia-wyjścia; Segmenty i procedury; Funkcje wewnętrzne; Przykłady zastosowań; Kompilacja, konsolidacja i wykonywanie programów w systemie MS DOS; Dodatek — repetytorium składni; Wykaz konstrukcji składniowych.

Książka jest przeznaczona dla początkujących i zaawansowanych czytelników interesujących się zagadnieniami oprogramowania mikrokomputerów.

WARUNKI PRENUMERATY NA LATA 1988—1989

Prenumeratorzy zbiorowi — jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłaty wyłącznie na blankiecie „wpłata—zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia).

Blankiety te będą dostarczane dotychczasowym prenumeratorom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemne lub telefoniczne) w Zakładzie Kolportażu.

Prenumeratorzy indywidualnie — osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: NBP III Oddział Warszawa 1036-7490-139-11.

Prenumerata ulgowa — przysługuje wyłącznie osobom fizycznym — członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po 1 egzemplarzu każdego czasopisma.

Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumerata ze zleceniem wysyłki za granicę — zamawia się tak jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy.

Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach:

- do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny,
- do 28 lutego na II, III i IV kwartał oraz II półrocze,
- do 31 maja na III i IV kwartał oraz II półrocze,
- do 31 sierpnia na IV kwartał.

Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela — Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 wew. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism — można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 27-43-65), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena w 1988 r. (dotyczy numerów 7—12)

Cena egzemplarza		Prenumerata			
		kwartalna		półroczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
250 zł	50 zł	750 zł	150 zł	1500 zł	300 zł

Cena w 1989 r.

Cena egzemplarza		Prenumerata					
		kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
300 zł	60 zł	900 zł	180 zł	1800 zł	360 zł	3600 zł	720 zł

MICHAŁ BARTYŚ
Instytut Automatyki Przemysłowej
Warszawa

Procedury INTR i MSDOS w MS-Pascalu

W artykule przedstawiono podstawowe zasady łączenia i przekazywania parametrów aktualnych między programami napisanymi w języku asemblera MASM i języku wysokiego poziomu MS-Pascal. Zasady te zilustrowano na przykładzie implementacji niestandardowych procedur INTR i MSDOS w MS-Pascalu.

Jedną z istotnych zalet systemu operacyjnego DOS jest możliwość wykorzystania w programach użytkowych jego rozlicznych wewnętrznych funkcji systemowych [3, 8, 9]. Wywołanie dowolnej funkcji systemu DOS polega na realizacji tzw. programowego przerwania systemowego o określonym identyfikatorze i parametrach. Zarówno identyfikator jak i parametry przerwania są przekazywane przez tzw. rejestry DOS [3] zgodne co do liczby i oznaczeń z rejestrami mikroprocesorów rodziny 8088/8086.

W odróżnieniu od Turbo Pascala [1, 2] w MS-Pascalu [6] nie przewidziano zasadniczo możliwości bezpośredniego wykorzystania funkcji i procedur DOS, a zakres działania właściwej dla MS-Pascala funkcji DOSXQQ jest bardzo ograniczony. Użytkownik pragnący wykorzystać w programie pascalowym pełny wachlarz funkcji systemu DOS musi zatem uciec się do dołączenia do własnego programu procedur lub funkcji, przygotowujących parametry przerwania programowego DOS, bądź do napisania własnych odpowiedników procedur INTR czy MSDOS znanych z Turbo Pascala. Zarówno w jednym, jak i drugim wypadku, programista musi respektować określone zasady łączenia programów napisanych w języku asemblera i języku wysokiego poziomu.

Przedstawione procedury INTR i MSDOS w MS-Pascalu, deklarowane jako zewnętrzne, można dołączyć na etapie konsolidacji do innych modułów wynikowych wchodzących w skład programu użytkowego. Użycie tych procedur otwiera także drogę do wykorzystania bibliotek funkcji DOS, np. [4].

ZASADY ŁĄCZENIA PROGRAMÓW W JĘZYKU ASEMBLERA I MS-PASCALU

Zasady łączenia programów napisanych w języku asemblera MASM [5] i w języku MS-Pascal [6] opisano między innymi w [7 9]. Z praktycznego punktu widzenia zasady te można sklasyfikować w następujący sposób:

- zasady deklarowania funkcji i procedur,
- zasady ochrony zawartości rejestrów i stosu,
- zasady przekazywania parametrów aktualnych i wartości funkcji.

Dr inż. MICHAŁ BARTYŚ ukończył w 1973 r. Wydział Mechaniki Precyzyjnej Politechniki Warszawskiej, a w 1983 r. uzyskał tytuł doktora. Od 1973 r. pracuje w Instytucie Automatyki Przemysłowej Politechniki Warszawskiej, zajmując się zagadnieniami związanymi z automatyzacją eksperymentalnych badań maszyn.

Tabela 1. Zasady łączenia programów napisanych w języku asemblera MASM i w języku MS-Pascal

Zasady deklarowania funkcji i procedur
1. Nazwy funkcji i procedur napisanych w języku asemblera powinny być zadeklarowane jako funkcje zewnętrzne (z dyrektywą EXTERN). Przykład: PROCEDURE ADD (VAR A, B, C: INTEGER); EXTERN;
2. Funkcje i procedury w programie napisanym w języku asemblera, przeznaczone do wykorzystania w programie pascalowym, powinny być zadeklarowane jako ogólnie dostępne (z dyrektywą PUBLIC). Przykład: PUBLIC ADD ... ADD PROC FAR
3. Zmienne i stałe powinny być zadeklarowane w segmencie DATA z nazwą 'DATA'.
4. Procedura lub funkcja powinna być zadeklarowana w segmencie o nazwie 'CODE'.
5. Rejestrom CS, DS i SS należy przyporządkować, odpowiednio, nazwy segmentu programu i segmentu danych
Zasady zachowania zawartości rejestrów i stosu
6. W wyniku wykonania funkcji lub procedury napisanej w języku asemblera nie mogą ulec zmianie zawartości rejestrów: BP, DS, SS. Jeśli funkcje lub procedury korzystają z tych rejestrów to pierwotne zawartości tych rejestrów muszą być odtworzone tuż przed zakończeniem wykonywania procedury lub funkcji. Podobnie, wszystkie zmienne umieszczone na stosie przez procedurę lub funkcję wywołującą muszą być przed jej zakończeniem usunięte ze stosu.
Zasady przekazywania parametrów oraz wartości funkcji i procedur

7. Wywołanie funkcji lub procedury pascalowej z dyrektywą EXTERN powoduje, że parametry aktualne procedury lub funkcji są ładowane na stos w kolejności ich zadeklarowania. Procedura lub funkcja napisana w języku asemblera, w końcowej fazie realizacji, powinna odtworzyć wskaźnik BP oraz usunąć ze stosu wszystkie załadowane w trakcie wywołania procedury lub funkcji pascalowej parametry aktualne oraz dodatkowo dwubajtowy adres struktury, jeśli funkcja jest typu strukturalnego.
Poniżej przedstawiono typowy ciąg instrukcji poprzedzających i kończących procedurę napisaną w języku symbolicznym.
 	
PUSH BP ; Zapamiętaj
MOV BP, SP ; wskaźnik BP
. . . ciało procedury
MOV SP, BP ; Odtwórz
POP BP ; wskaźniki BP i SP
RET N ; N — liczba bajtów pola parametrów (porównaj z rys. 1).

Zasady deklarowania funkcji i procedur definiują wymogi formalne (składniowe) deklaracji funkcji i procedur pascalowych i w języku symbolicznym MASM. Zasady ochrony zawartości rejestrów i stosu obejmują zestaw reguł służących ochronie zawartości określonych rejestrów i stosu. Zasady przekazywania parametrów aktualnych oraz wartości funkcji określają sposoby dostępu do wspólnego obszaru pamięci zmiennych i stałych przez programy: pascalowy i asemblerowy.

Szczegółowy wykaz zasad łączenia programów pascalowych i asemblerowych przedstawiono w tabeli 1.

ZASADY PRZEKAZYWANIA PARAMETRÓW AKTUALNYCH W IMPLEMENTACJI MS-PASCAL

Każdorazowemu wywołaniu procedury lub funkcji pascalowej towarzyszy zapisanie na stosie określonej struktury danych o ściśle sformalizowanej postaci. Znajomość konstrukcji tej struktury, a ściślej zawartości stosu, ma podstawowe znaczenie dla zrealizowania operacji łączenia programów pascalowych z programami napisanymi w języku asemblera.

Wywołanie **procedury pascalowej** z dyrektywą EXTERN powoduje, że parametry aktualne procedury są zapisywane na stosie w kolejności ich zadeklarowania. Jeśli parametr formalny jest deklarowany przez wartość, to na stos ładowana jest aktualna wartość tego parametru. Jeśli parametr został zadeklarowany przez adres jako VAR lub CONST, to na stos podawany jest dwubajtowy adres względny tego parametru (wyrównanie, ang. offset) w obrębie standardowego segmentu danych.

Jeśli parametr jest zadeklarowany przez adres jako VARS lub CONST, to na stos podawany jest czterobajtowy adres parametru w postaci segment:wyrównanie, przy czym jako pierwszy jest zapisywany adres segmentu (2 bajty), a następnie adres względny w obrębie segmentu (2 bajty).

Bezpośrednio za ostatnim parametrem listy na stos jest przekazywany czterobajtowy bezwzględny adres powrotny, przy czym jako pierwszy przekazywany jest adres segmentu, a następnie adres względny w obrębie segmentu.

Wywołanie **funkcji pascalowej** z dyrektywy EXTERN powoduje utworzenie na stosie identycznej struktury danych jak w wypadku wywołania procedury pascalowej, o ile funkcja jest typu BYTE, CHAR, WORD, INTEGER2, ADR, INTEGER4 lub ADS. Po wykonaniu wartości funkcji jest przekazywana do rejestrów DX:AX. Obowiązują przy tym następujące reguły:

- dla funkcji typu BYTE i CHAR jej wartość jest przekazywana do rejestru AL,
- dla funkcji typu WORD, INTEGER2 i ADR jej wartość jest przekazywana do rejestru AX, przy czym bardziej znaczący bajt jest ładowany do rejestru AH, a mniej znaczący do rejestru AL,
- dla funkcji INTEGER4 i ADS bardziej znaczące słowo (adres segmentu w wypadku funkcji typu ADS) jest ładowane do rejestru DX, natomiast mniej znaczące (adres względny) — rejestru AX.

Tabela 2. Szkic zawartości stosu przed wykonaniem procedury DODAJ
Oznaczenia: SB — bardziej znaczący bajt, MB — mniej znaczący bajt, SEG ADR — adres segmentu, OFF ADR — adres względny, SP — wskaźnik stosu

Typ informacji	Bajt	Znaczenie
Integer	SB	A
Integer	MB	
OFF ADR	SB	B
OFF ADR	MB	
SEG ADR	SB	SUMA
SEG ADR	MB	
OFF ADR	SB	
OFF ADR	MB	
SEG ADR	SB	Adres powrotny
SEG ADR	MB	
OFF ADR	SB	
OFF ADR	MB	

W wypadku gdy funkcja jest innego typu, kompilator języka MS-Pascal ładuje dodatkowo na stos, bezpośrednio po ostatnim parametrze formalnym i przed adresem powrotnym, dwubajtowy adres względny w obrębie standardowego segmentu danych zmiennej strukturalnej będącej wartością funkcji. Po wykonaniu funkcji adres ten znajduje się w rejestrze AX.

W tabeli 2 przedstawiono schematycznie zarys stosu po wywołaniu przykładowej procedury pascalowej:

```
PROCEDURE DODAJ(
    A           : INTEGER;
    VAR B       : INTEGER;
    VAR SUMA    : INTEGER); EXTERN;
```

IMPLEMENTACJA PROCEDUR INTR I MSDOS W MS-PASCALU

Procedury INTR i MSDOS umożliwiają przekazywanie wartości parametrów aktualnych między programem pascalowym a funkcjami systemowymi, przy wykorzystaniu rejestrów DOS. Wzorem procedury INTR z Turbo Pascala można zdefiniować typ rekordowy z wariantami, z logicznym typem znacznikowym, odwołujący się do rejestrów DOS:

```
REG_TYPE = RECORD
CASE BOOLEAN OF
  TRUE : ( AX, BX, CX, DX, BP, SI, DI, DS, ES, FLAGS :
  : WORD );
  FALSE : ( AL, AH, BL, BH, CL, CH, DL, DH : BYTE );
END;
```

oraz podać deklaracje procedur INTR i MSDOS:

```
PROCEDURE INTR ( NUMBER : WORD; VARS REG :
  : REG_TYPE );
PROCEDURE MSDOS( VARS REG : REG_TYPE );
BEGIN
  INTR( 16#21, REG );
END;
```

```
;*************************************************************
;*                  DEKLARACJE PASCALOWE:                    *
;*-----------------------------------------------------------*
;*  DEKLARACJA TYPU  :                                       *
;*  ------------------                                       *
;*  REG_TYPE = RECORD                                        *
;*  CASE BOOLEAN OF                                          *
;*    TRUE  : (AX,BX,CX,DX,BP,SI,DI,DS,ES,FLAGS : WORD);     *
;*    FALSE : (AL,AH,BL,BH,CL,CH,DL,DH          : BYTE);     *
;*  END;                                                     *
;*-----------------------------------------------------------*
;*  DEKLARACJA PROCEDURY :                                   *
;*  ----------------------                                   *
;*                                                           *
;*  PROCEDURE INTR( NUMBER:WORD; VARS REG:REG_TYPE);         *
;*                                                           *
;*************************************************************


.8086

CODE        SEGMENT PARA   PUBLIC   'CODE'
PUBLIC      INTR
            ASSUME  CS:CODE

INTR        PROC    FAR

            ;*************************************************
            ;*                    I N T R                    *
            ;*-----------------------------------------------*
            ;* Procedura obslugi programowych przerwan systemu DOS *
            ;*                                               *
            ;*************************************************

            PUSH    BP              ; Zapamietaj wskaznik BP
            MOV     BP,SP           ; Ustaw nowa wartosc  BP
            PUSH    DS              ; Zapamietaj rejestr  DS
            PUSH    BP              ; Zapamietaj wskaznik SP

            PUSH    SS              ; Zapamietaj zawartosci pozostalych
            PUSH    AX              ; rejestrow
            PUSH    BX              ;
            PUSH    CX              ;
            PUSH    DX              ;
            PUSH    SI              ;
            PUSH    DI              ;
            PUSH    ES              ;
            PUSHF                   ;

            MOV     AX,10[BP]       ; Odczytaj numer przerwania programowego
            CMP     AX,03H          ; Czy to jest INT 3 ?
            JE      CREATE_INT_3    ; Tak to INT 3. Utworz kod przerwania
            MOV     AH,0CDH         ; Wprowadz do AH kod instrukcji INT 0
            JMP     CREATE_ALL      ; Kontynuuj tworzenie kodu przerwania
CREATE_INT_3: MOV   AX,0CC90H       ; Wprowadz do AX kod przerwania INT 3
CREATE_ALL  : XCHG  AH,AL           ; Zamien bajty
            LEA     DI,INTER_RUPT   ; Odczytaj adres instrukcji INT
            MOV     CS:[DI],AX      ; Zapisz kod instrukcji     INT
            JMP     NEXT_INST       ; Instrukcja pusta

NEXT_INST:  MOV     DI,6[BP]        ; Odczytaj adres segmentu rekordu    REG
            MOV     ES,8[BP]        ; Odczytaj adres wzgledny rekordu    REG
            MOV     BX,0            ; Wyzeruj wskaznik elementow rekordu REG
```

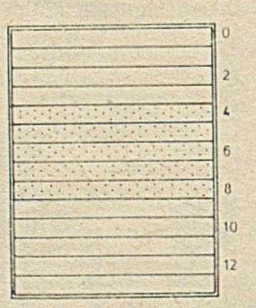
```
PUSH_REG:   CMP     BX,OAH+OAH      ; Rekord REG ma dlugosc 20 bajtow
            JE      POP_REG         ; Czy koniec operacji na stosie ?
            PUSH    ES:[DI+BX]      ; Zapisz kolejny element rekordu na stos
            ADD     BX,2            ; Inkrementacja wskaznika elementu  REG
            JMP     PUSH_REG        ; Powtorz operacje zapisu REG na stosie

POP_REG:    POPF                    ; Zapisz rekord REG do odpowiednich rej.
            POP     ES              ;
            POP     DS              ;
            POP     DI              ;
            POP     SI              ;
            POP     BP              ;
            POP     DX              ;
            POP     CX              ;
            POP     BX              ;
            POP     AX              ;

            STI                     ; Odblokuj przerwania
INTER_RUPT: NOP                     ; Wykonaj przerwanie DOS.
            NOP                     ; Drugi bajt rezerwy na kod przerwania

            PUSHF                   ; Zapisz na stos zawartosci rejestrow po
            PUSH    ES              ; wykonaniu przerwania
            PUSH    DS              ;
            PUSH    DI              ;
            PUSH    SI              ;
            PUSH    BP              ;
            PUSH    DX              ;
            PUSH    CX              ;
            PUSH    BX              ;
            PUSH    AX              ;

            MOV     BX,SP           ; Odczytaj wskaznik stosu SP
            MOV     DS,SS:40[BX]    ; Odtworz  rejestr  DS
            MOV     BP,SS:38[BX]    ; Odtworz  wskaznik BP

            MOV     SI,6[BP]        ; Odczytaj adres wzgledny rekordu   REG
            MOV     ES,8[BP]        ; Odczytaj adres segmentu rekordu   REG
            MOV     BX,0            ; Wyzeruj wskaznik elementow rekordu REG

RETURN_REG: CMP     BX,OAH+OAH      ; Rekord REG ma dlugosc 20 bajtow
            JE      FINISH          ; Czy koniec zapisu ze   stosu do  REG ?
            POP     ES:[SI+BX]      ; Zapisz kolejny element stosu do  REG
            ADD     BX,2            ; Inkrementacja wskaznika elementu REG
            JMP     RETURN_REG      ; Powtorz operacje zapisu stosu do REG

FINISH :    POPF                    ; Odtworz pierwotne zawartosci rejestrow
            POP     ES              ; procesora
            POP     DI              ;
            POP     SI              ;
            POP     DX              ;
            POP     CX              ;
            POP     BX              ;
            POP     AX              ;
            POP     SS              ;

            POP     BP              ; Odczytaj wskaznik SP do rejestru BP
            POP     DS              ; Odtworz rejestr  DS
            MOV     SP,BP           ; Odtworz wskaznik SP
            POP     BP              ; Odtworz wskaznik BP
            RET     6               ; Koniec procedury z przeskokiem
                                    ; pola parametrow
INTR        ENDP

CODE        ENDS
```

Wydruk 1. Implementacja procedury INTR w MS-Pascalu

gdzie zmienna NUMBER jest numerem programowego przerwania systemowego DOS.

Procedura MSDOS jest więc równoważna procedurze INTR przy założeniu, że wartość parametru aktualnego NUMBER jest stała i wynosi 21H. Procedurę tę można również utworzyć w sposób analogiczny do procedury INTR (por. z wydrukiem 1) wprowadzając niewielkie modyfikacje kodu.

Jak wspomniano na wstępie, deklaracja procedury INTR w programie użytkowym musi być dokonana z dyrektywą EXTERN, chyba że zostanie ona przez użytkownika dołączona do biblioteki PASCAL.LIB. Zwykle procedurę tę dołącza się na etapie konsolidacji do innych modułów wynikowych wchodzących wskład programu użytkowego (por. wydruk 2). Proponowana procedura INTR nie korzysta z obszaru pamięci danych, nie zmienia zawartości rejestrów programu i przekazuje, bezpośrednio przed wykonaniem programowego przerwania systemowego, stany rejestrów DOS zgodnie z aktualną zawartością rekordu REG. Ponadto, bezpośrednio po wykonaniu programowego przerwania systemowego przekazuje stany rejestrów DOS do rekordu REG, zgodnie z ich stanem aktualnym w momencie zakończenia wykonywania przerwania.



Szkic kształtu kursora na ekranie monitora mikrokomputera kompatybilnego z IBM PC, wyposażonego w monograficzną kartę grafiki Hercules; parametry kształtu kursora: — położenie górnej krawędzi CURSOR.UPPER = 4, — położenie dolnej krawędzi CURSOR.LOWER = 8

W celu zilustrowania sposobu wykorzystania procedury INTR na wydruku 2 podano przykłady dwóch procedur pascalowych.

```
PROGRAM PRZYKLAD_UZYCIA_PROCEDURY_INTR (INPUT,OUTPUT);
TYPE
  REG_TYPE=RECORD
  CASE BOOLEAN OF
    TRUE: (AX,BX,CX,DX,BP,SI,DI,DS,ES,FLAGS : WORD);
    FALSE:(AL,AH,BL,BH,CL,CH,DL,DH          : BYTE);
  END;

  CURSOR_TYPE = RECORD
     UPPER       : BYTE;
     LOWER       : BYTE;
     X           : BYTE;
     Y           : BYTE;
     PAGE_NUMBER : BYTE;
  END;

PROCEDURE INTR( NUMBER : WORD; VARS REG : REG_TYPE ); EXTERN ;

PROCEDURE SET_CURSOR_SHAPE
        (VAR CURSOR : CURSOR_TYPE);
{*********************************************************
** USTAWIA KSZTALT KURSORA.                             **
** ZMIENNE : CURSOR.UPPER ORAZ CURSOR.LOWER MOGA PRZYJ- **
** MOWAC DOWOLNE WARTOSCI CALKOWITE W ZAKRESIE (0 - 127). **
*********************************************************}
VAR
  REG : REG_TYPE;

BEGIN (* SET_CURSOR_SHAPE *)
    REG.AH := 1;
    REG.CH := CURSOR.UPPER;
    REG.CL := CURSOR.LOWER;
    INTR(16#10,REG);
END;  (* SET_CURSOR_SHAPE *)

PROCEDURE GET_CURRENT_DISK
        (VAR CURRENT_DISK : INTEGER);
{*********************************************************
** PODAJE KOD CYFROWY  DYSKU AKTUALNEGO.                **
** DYSK  A ----> KOD  0    ;  DYSK  B ----> KOD  1      **
** DYSK  C ----> KOD  2    ;  ITD.                      **
*********************************************************}
VAR
  REG       : REG_TYPE;

BEGIN (* GET_CURRENT_DISK *)
  REG.AH      := WRD(16#19);
  INTR(16#21,REG);  (* CURRENT DISK *)
  CURRENT_DISK := ORD(REG.AL);
END;  (* GET_CURRENT_DISK *)

BEGIN END.
```

Wydruk 2. Ilustracja użycia procedury INTR w programie pascalowym

Pierwsza procedura umożliwia dokonanie zmiany kształtu kursora na ekranie monitora; może być również użyta do usunięcia kursora z ekranu. Zmienna strukturalna CURSOR definiuje następujące parametry:

- współrzędne kursora na ekranie — X, Y,
- numer aktualnej strony kursora — PAGE_NUMBER,
- położenie górnej i dolnej krawędzi prostokąta kursora (por. rys.) — UPPER, LOWER.

Zmiana wartości CURSOR.UPPER i CURSOR.LOWER wywołuje zmianę kształtu kursora. Kursor staje się niewidoczny, gdy dla monograficznej karty grafiki wartości CURSOR.UPPER jest większa niż CURSOR.LOWER, lub gdy obie wartości są równe. Użycie funkcji INTR umożliwia także realizację innych operacji na kursorze, takich jak: zmianę jego pozycji, zapamiętanie bieżącej pozycji, odtworzenie jego wcześniej zapamiętanego kształtu i pozycji itp.

Druga procedura umożliwia uzyskanie informacji o numerze dysku. Procedura ta korzysta z funkcji DOS 21H o parametrze 19H. Użycie procedury INTR w tym wypadku jest więc równoważne użyciu funkcji MSDOS z parametrem 19H. W wyniku jej wykonania, do zmiennej CURRENT_DISK zostaje zapisana zakodowana informacja o numerze dysku. Sposób kodowania ilustruje komentarz podany na wydruku 2.

Jest oczywiste, że zaprezentowane wyżej przykłady, ilustrujące głównie technikę wykorzystania procedury INTR, nie mogą być uważane za reprezentatywne dla istniejących już i odpowiednio obszernych bibliotek funkcji DOS [4].

dokończenie na s. 18

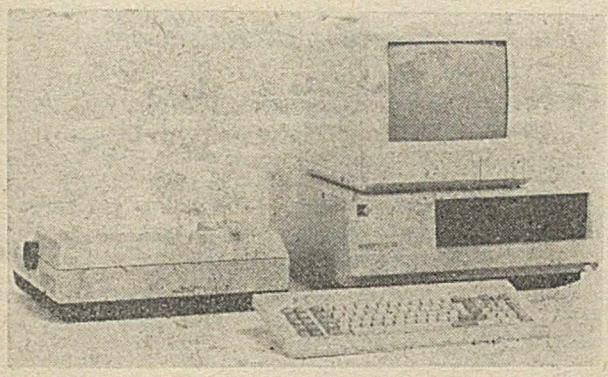
KRZYSZTOF DZIK
Instytut Maszyn Matematycznych
Warszawa

Mazovia 1016 — konfiguracja sprzętowa

Mazovia 1016 jest profesjonalnym mikrokomputerem osobistym, w którym konsekwentnie wprowadzono polskie znaki. Użytkownik, wykorzystując polską klawiaturę, może wyświetlać na ekranie, bądź drukować na drukarce teksty zawierające polskie litery. "Polonizacji" sprzętu towarzyszy zestaw typowych programów użytkowych, zwłaszcza z zakresu:

— procesorów tekstowych,
— baz danych,
— arkuszy rachunkowych,
— zastosowań graficznych.

Programy te komunikują się z użytkownikiem po polsku i wykonują niektóre funkcje, np. sortowanie, według polskiego alfabetu.



Mazovia 1016

Mikrokomputer Mazovia 1016 składa się z następujących urządzeń (fot.):

— jednostka centralna,
— klawiatura KL10,
— monitor monochromatyczny MM 12P,
— drukarka D 100E/PC.

KLAWIATURA KL-10

Klawiatura KL-10 może także współpracować z dowolnym komputerem kompatybilnym z IBM PC lub IBM PC/XT. Dołączenie jej do wyżej wymienionych komputerów nie wymaga żadnych adaptacji. Sprzęg klawiatury zapewnia pełną swobodę definiowania przez oprogramowanie systemowe znaczenia poszczególnych klawiszy. Osiągnięto to dzięki rezygnacji z generowania przez klawiaturę standardowego kodu (np. ASCII). Klawiatura generuje 8-bitowy kod położeniowy, tzw. skankod, który określa numer klawisza i czynność — wciśnięcie lub zwolnienie klawisza. Skankod generowany przy zwolnieniu klawisza jest tworzony przez dodanie 80H do wartości skankodu danego klawisza, generowanego przy wciśnięciu.

Klawiatura KL-10 ma 85 klawiszy zgrupowanych w trzech sekcjach. W sekcji centralnej znajdują się klawisze normalnej maszyny do pisania (wg PN F/2000). W lewej sekcji znajduje się 10 klawiszy funkcyjnych (F1—F10), których sposób użycia jest definiowany przez oprogramowanie. W sekcji kalkulatora po prawej stronie znajduje się 18 klawiszy — w tym 5 sterujących. Jeden z nich (kPLz) służy do zmiany trybu pracy klawiatury. Użycie pozostałych 13 klawiszy, definiowane przez oprogramowanie, uwzględnia istniejące opisy dotyczące wpisywania cyfr lub sterowania kursorem i jego wydrukiem.

W celu przystosowania klawiatury do potrzeb polskiego użytkownika wprowadzono trzy tryby pracy: **tryb IBM** (standardowy, zgodny ze stosowanym w komputerze PC/XT), **kPL** (polski) i **PLz** (tryb polski złożeniowy).

W trybie IBM, 83 klawisze (spośród 85) reprezentują standardową klawiaturę współpracującą z komputerem PC/XT, zgodnie z opisem na bocznych powierzchniach klawiszy. Dla trybów kPL i PLz obowiązują opisy na górnych powierzchniach. Znak z prawego górnego roku klawisza uzyskuje się przez jednoczesne wciśnięcie **ALT3** i

Dane techniczne drukarki D-100 PC

Parametr	Wartość
Sposób drukowania	Uderzeniowy, matrycowy
Szybkość drukowania	100 znaków/s przy 10 znakach/cal
Kierunek drukowania	Dwukierunkowy dla tekstów, jednokierunkowy dla druku graficznego oraz programowany
Głowica drukująca	9 igieł, średnica igieł 0,29 mm
Odległość między wierszami	1/6 cala, 1/8 cala lub programowany mikrokrok 1/216 cala
Matryca druku	11 × 9 punktów
Zestaw znaków	96 znaków ASCII, 96 znaków kursywy ASCII, międzynarodowe zestawy znaków: nr 1 I-PCL (dla PC), nr 2 I-POL/ITALIC (dla PC)
Opcjonalne zestawy znaków	LILIVEST, IHROM, CS, WEST, RUS GRAPHIC
Rodzaje druku	Standard (10 znaków/cal), elite (12 znaków/cal), zagęszczony (16,5 znaku/cal), szeroki (5 znaków/cal), fazowy, podwójny szeroki zagęszczony, elite zagęszczony, super/subscript, kursywa
Wymiary znaków	2,10 × 3,10 mm — standard 1,05 × 3,10 mm — zagęszczony 4,20 × 3,10 mm — szeroki 1,40 × 3,10 mm — elite 2,80 × 3,10 mm — elite zagęszczony 2,10 × 1,60 mm — super/subscript
Długość wiersza	80 znaków — standard 40 znaków — szeroki 135 znaków — zagęszczony 68 znaków — szeroki zagęszczony 96 znaków — elite 48 znaków — elite zagęszczony
Nośnik informacji	Papier obustronnie perforowany o stałej szerokości i rozstawie otworów 238 mm (9 3/8 cala, z tolerancją ±1,6 mm) lub 228 mm z tolerancją ±1,6 mm. Papier obustronnie perforowany o zmiennej szerokości od 4 cali do 9 3/8 cala (opcja). Pojedyncze arkusze papieru formatu A4 (opcja).
Liczba kopii	1+2
Niezawodność	1500 h — MTBF (wyłączając głowicę); 100 mln znaków dla głowicy
Sprzęgi	Równoległy CENTRONICS, szeregowy V24 (RS 232C, S2) równoległy + szeregowy
Taśma barwiąca	Czarna, 16 m długości, zamknięta w pętlę Moebiusa, dostarczana w postaci wymiennych ładunków do kasety (RIBPACK 1 nr 66KPO350-01); ładunek firmy EPSON „EPSON MX100 ZIPPACK" lub podobne

danego klawisza. Przy wykorzystaniu klawisza **ALT3** uzyskuje się duże litery polskiego alfabetu. Na przykład, w celu uzyskania litery ś należy wcisnąć klawisz **ALT3**, a następnie jednocześnie klawisze **Shift** i s. Za jego pomocą można też generować litery ze znakiem dierezy (niem. umlaut), np. w celu uzyskania litery ä należy wcisnąć jednocześnie **ALT3** i klawisz **12**, a następnie literę **a**.

W trybie kPL klawiatura generuje ciągi sygnałów, które obsłużone przez standardowy BIOS komputera PC/XT dają rozszerzone kody ASCII (ang. Extended ASCII), reprezentujące dane znaki narodowe. Kody te są akceptowane poprawnie przez sterownik monitora mikrokomputera Mazovia 1016 i drukarkę. Inne sterowniki monitorów i drukarki wymagają przeprogramowania generatora znaków w obszarze znaków narodowych, gdyż nie ma zakodowanych w nich znaków charakterystycznych dla polskiego alfabetu.

Tryb PLz nie wymaga przeprogramowania generatorów znaków, gdyż polskie znaki są tworzone ze złożenia: znak modyfikujący, BS (cofnięcie), litera, np. **Ł** jest tworzona z ciągu: **[/] [BS] [L]**. Na ekranie monitora znak modyfikujący jest niewidoczny, gdyż jest on bardzo krótko wyświetlany, a następnie pojawia się litera. Natomiast podczas drukowania cały znak zostanie odwzorowany prawidłowo.

Po wyłączeniu zasilania lub wyzerowaniu komputera, klawiatura jest ustawiana w trybie pracy IBM. Jeżeli istnieje konieczność wykorzystywania polskich liter, należy przełączyć klawiaturę w tryb kPL. Dokonuje się tego wciskając jednocześnie trzy klawisze: **CTRL**, **ALT** i **kPLz/IBM**. Operacja ta nie generuje skankodu, a jedynie steruje wewnętrzną pracą klawiatury.

Przejście do trybu PLz jest sygnalizowane przez diodę o znaczeniu PLz. Przejście do kolejnego trybu pracy uzyskuje się w identyczny sposób. Tryb IBM jest sygnalizowany przez diodę LED o znaczeniu iBM (w trybie pracy kPL obydwie diody są zgaszone). Oprócz diod wskazujących tryb pracy klawiatury istnieją jeszcze trzy inne: **NUM LOCK** — wskazująca przełączenie prawej sekcji klawiatury w tryb kalkulatorowy, **CAPS LOCK** — sygnalizująca uaktywnienie górnego zestawu znaków, i **SCROLL LOCK** — wskazująca użycie klawiszy sterowania kursorem.

Klawiatura KL-10 jest zbudowana z folii pojemnościowej i mikroprocesora Intel 8035. Do zadań mikroprocesora należy kontrola stanu klawiszy i linii transmisyjnych z i do jednostki centralnej, przetwarzanie dopasowujące narodowy rozkład klawiszy do wymagań standardu IBM PC, oraz sterowanie świeceniem diod. Dodatkowym zadaniem jest (w wypadku włączenia zasilania lub na żądanie jednostki centralnej) kontrola własnej pamięci RAM, pamięci EPROM oraz sprawdzenie czy jakiś klawisz nie został zablokowany w pozycji „wciśnięty".

Klawiatura jest dołączona do modułu jednostki centralnej kablem przez łącze DIN (magnetofonowe). Jest to kabel ekranowy zawierający przewód zasilający, masę oraz dwie dwukierunkowe linie sygnałowe. Kabel jest skręcony w spiralę, podobnie jak przewód słuchawki telefonicznej. Klawiatura ma dwa położenia pracy, normalne i uniesione o około 7°, po odchyleniu pary nóżek.

MONITOR EKRANOWY MM12P

Monitor monochromatyczny MM12P spełnia w mikrokomputerze Mazovia 1016 funkcję standardowego wyświetlacza informacji, przygotowanej do zobrazowania zarówno w sposób monochromatyczny jak i kolorowy, znakowy i graficzny. Monitor może być także dołączany do innych sterowników i mikrokomputerów zgodnych ze standardem IBM PC.

W wypadku dołączenia do sterownika zgodnego z IBM Monochrome Display/Printer Adapter lub Monochrome Graphics/Printer Adapter monitor zapewnia pracę z rozdzielczością 720 punktów × 350 linii w trybie graficznym lub 80 kolumn × 25 wierszy w trybie znakowym, z matrycą 9 × 14 punktów. W połączeniu ze sterownikiem zgodnym z IBM Color Graphica Adapter monitor zapewnia wyświetlenie tekstu i grafiki w 16 odcieniach szarości z rozdzielczością 640 punktów × 200 linii.

DRUKARKA D-100 PC

Drukarka D-100 PC jest drukarką mozaikową, uderzeniową, przeznaczoną do współpracy z systemami komputerowymi w celu wprowadzania informacji w postaci druku alfanumerycznego lub graficznego. Ma głowicę uderzeniową 9-igłową, która podczas druku kształtuje znak w matrycy 11 × 9 punktów z szybkością 100 znaków/s.

Drukarka może pracować w dwóch trybach pracy, oznaczonych symbolami **E** i **PC**. W trybie **E** realizuje funkcje drukarki FX-80 firmy EPSON, a w trybie PC — funkcje drukarki IBM Graphics Printer oraz dodatkowo akceptuje większość rozkazów sterujących drukarki EPSON FX-80.

Drukarka D100 PC może być wyposażona w sprzęg równoległy Centronics lub szeregowy RS 232 (lub też w obydwa jednocześnie). Linie sprzęgu Centronics są wyprowadzone na 37-stykowe złącze szufladowe — wtyk ELTRA 871037, natomiast linie RS 232 — na 25 stykowe złącza szufladowe — wtyk ELTRA 871-02502211-001..

Drukarka umożliwia wydruk na papierze perforowanym o szerokości 250 mm (lub 240 mm na specjalne zamówienie) lub pojedynczych kartkach papieru formatu A4. W wypadku drukowania na pojedynczych kartkach zaleca się stosowanie podajnika kart. Dane techniczne drukarki przedstawiono w tabeli.

Ceny ogłoszeń

Od 1 lipca 1987 r. obowiązują następujące ceny materiałów reklamowych publikowanych na łamach INFORMATYKI:

Ogłoszenia

— ogłoszenia czarno-białe, artykuły reklamowe i informacje naukowo-techniczne (biuletyny) zależnie od objętości: cała strona — 50 tys., 3/4 str. — 45 tys., 2/3 str. — 40 tys., 1/2 str. — 35 tys., 1/3 str. — 30 tys., 1/4 str. — 25 tys., 1/8 str. — 20 tys., poniżej 1/8 str. — 200 zł za cm²,
— ogłoszenia drobne (zależnie od liczby słów) jedno słowo — 50 zł.

Dodatki do ceny podstawowej:

— za każdy dodatkowy kolor + 30%,
— za każdy specjalny kolor (nie wynikający z podstawowych kolorów) + 30%,
— za pełny kolor (grafika wielobarwna, zdjęcia kolorowe) + 120%
— za zamieszczenie ogłoszenia na I lub IV stronie okładki + 100%
— za zamieszczenie ogłoszenia na II i III stronie okładki + 50%

Zniżki

dotyczą ogłoszeń — całkowitych powtórzeń
— za ogłoszenia 3—5-krotne — 10%
— za ogłoszenia 6—10-krotne — 20%
— za ogłoszenia 11-krotne i powyżej — 30%
— za artykuły i wkładki reklamowe wykonane przez zleceniodawcę — 40%
— za biuletyny i bloki reklamowe — 60%

W uzasadnionych wypadkach stosuje się z niżki specjalne dla ogłoszeń nie będących powtórzeniami — za zgodą Dyrektora — Naczelnego Redaktora Wydawnictw NOT SIGMA.

Ponadto Biuro Ogłoszeń świadczy usługi w zakresie wykonywania zdjęć czarno białych i barwnych oraz nadbitek wkładek reklamowych.

Ceny wkładek

— wkładka 2 str. o formacie A4
nakład do 500 egz. 20 tys. zł
nakład 500—1000 egz. 35 tys. zł
— wkładka 4 str. o formacie A4
nakład 500 egz. 40 tys. zł
nakład 500—1000 egz. 70 tys. zł

Przy wkładkach o nakładzie powyżej 1000 egz. stosuje się wielokrotność powyższych cen, a dodatki za kolory oblicza się jak dla ogłoszeń.

Ogłoszenia przyjmowane są przez:

Dział Ogłoszeń i Reklamy WCiKT NOT SIGMA
ul. Świętojerska 5/7, 00-236 Warszawa
adres do korespondencji: skrytka pocztowa 1004, 00-950 Warszawa
telefony: 31-93-65 lub 31-22-21 w. 196 i 291

JAN BIELECKI
Warszawa

Turbo C

Zbiory nagłówkowe, prototypy, deklarowanie funkcji

Zgodnie z proponowaną normą ANSI, każda funkcja biblioteczna jest zadeklarowana w pewnym zbiorze nagłówkowym. Zawartość tego zbioru jest włączana do programu za pomocą dyrektywy #include. Poza deklaracjami funkcji i zmiennych, zbiory nagłówkowe mogą zawierać także definicje typów i makrodefinicje. Przyjmuje się, że wszystkie identyfikatory zewnętrzne zadeklarowane w zbiorze nagłówkowych są zastrzeżone i mogą być używane jedynie w przypisanym im znaczeniu. Skutki dowolnych, nawet tożsamych, przedefiniowań takich identyfikatorów są zależne od implementacji.

Uwaga: W implementacji Turbo C są dozwolone tożsame przedefiniowania makrodefinicji.

Zbiory nagłówkowe mogą być włączane do programu wielokrotnie i w dowolnej kolejności. Mimo iż zbiór nagłówkowy nie musi być realizowany jako zbiór danych, zabrania się, aby w środowisku kompilacyjnym programu znajdowały się zbiory danych, które nie są zbiorami nagłówkowymi, a których nazwy kolidują z nazwami zbiorów nagłówkowych.

Uwaga: W implementacji Turbo C każdy ze zbiorów nagłówkowych jest zwykłym zbiorem danych.

Każda funkcja zadeklarowana w zbiorze nagłówkowym może być zrealizowana jako makrodefinicja biblioteczna albo funkcja biblioteczna. Jeśli w programie nie zostanie użyta dyrektywa #include, włączająca pewien zbiór nagłówkowy, to odwołania do funkcji zadeklarowanych w tym zbiorze będą realizowane jako odwołania do funkcji bibliotecznych. W przeciwnym razie, będą realizowane jako odwołania do makrodefinicji bibliotecznych. W takim wypadku zabrania się jawnego daklarowania takich funkcji.

Jeśli w kontekście, w którym wywołanie funkcji zostałoby zrealizowane za pomocą makrodefinicji bibliotecznej, jest pożądane odwołanie się do funkcji bibliotecznej, to można to zrealizować, posługując się dyrektywą #undef (anulującą makrodefinicję) albo ujmując nazwę wywoływanej funkcji w nawiasy okrągłe. W tym drugim wypadku nie dojdzie do zrealizowania makrowywołania, ponieważ bezpośrednio po nazwie funkcji nie wystąpi nawias okrągły otwierający.

W celu maksymalnego upodobnienia skutków użycia makrodefinicji i funkcji bibliotecznych, wymaga się takiego ich implementowania, aby każdy argument wywołania był opracowywany jednokrotnie, a makrodefinicja zawierała tak rozmieszczone nawiasy okrągłe, aby skutek odwołania do makrodefinicji bibliotecznej był taki sam, jak skutek odwołania do stowarzyszonej z nią funkcji bibliotecznej.

Przykłady

Deklaracja funkcji bibliotecznej printf znajduje się w zbiorze nagłówkowym stdio.h, a makrodefinicja biblioteczna abs znajduje się w zbiorze nagłówkowym stdlib.h.

A. Odwołanie do funkcji bibliotecznej abs

```
#include <stdio.h>
int abs();
main()
{
  printf("%d", abs(—13));
}
```

B. Odwołanie do makrodefinicji bibliotecznej abs

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  printf("%d", abs(—13));
}
```

C. Odwołanie do funkcji bibliotecznej abs

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  printf("%d", (abs) (—13));
}
```

D. Odwołanie do funkcji bibliotecznej abs

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  #undef abs
  printf("%d", abs(—13));
}
```

Deklaracje funkcji bibliotecznych występujące w zbiorach nagłówkowych mają zazwyczaj postać deklaracji prototypowych, w których rolę deklaratora funkcji pełni jej prototyp. Prototyp różni się tym od zwykłego deklaratora, że w nawiasach okrągłych, następujących bezpośrednio po identyfikatorze deklarowanej funkcji, jest zawarta lista deklaracji jej kolejnych parametrów. Identyfikatory parametrów zawarte w tej liście mogą być obrane dowolnie i nie kolidują z pozostałymi indentyfikatorami programu. Identyfikatory te mogą być opuszczone. W takim wypadku musi to dotyczyć wszystkich elementów listy.

Jeśli deklarowana funkcja może być wywoływana ze zmienną liczbę argumentów, to ostatnim elementem listy deklaracji może być symbol ... (trzy kropki). Użycie takiego symbolu nie dostarcza żadnej informacji o dopuszczalnych typach dodatkowych argumentów.

Jeśli deklarowana funkcja jest bezparametrowa, to rozpatrywana lista deklaracji upraszcza się do słowa kluczowego void.

Przykłady deklaracji prototypowych

A. Deklaracja bezparametrowej funkcji, której rezultat jest typu (char)

```
char Fun(void);
```

B. Deklaracja dwuparametrowej funkcji, o parametrach typu (int) i (char *), której rezultat jest typu (unsigned int)

```
unsigned Fun(int,char *);
```

C. Deklaracja jednoparametrowej funkcji, o parametrze typu (char), której rezultatem jest wskazanie dwuparametrowej funkcji o parametrach typu (int), udostępniającej rezultat typu (float)

```
float (*fun(char)) (int,int);
```

D. Deklaracja funkcji jak wyżej, w której prototypie, dla większej czytelności deklaracji, posłużono się identyfikatorami parametrów

```
float (*Fun(char Chr)) (int Day,int Week);
```

E. Deklaracja trójelementowej tablicy wskazań dwuparametrowych funkcji, o parametrach typu (char *) i (char (*) [5]), udostępniających rezultaty typu (int *)
int * (*Arr[3]) (char *,char (*) [5]);

F. Deklaracja jednoparametrowej funkcji, o parametrze typu (int), której rezultatem jest wskazanie jednoparametrowej funkcji, o parametrze typu (char), a która może być wywoływana co najmniej z jednym argumentem i udostępnia rezultat typu (int)
int (*fun(int)) (char Count, ...);

Deklaracje prototypowe mogą występować także poza zbiorami nagłówkowymi. W takim wypadku, jeśli dotyczą funkcji bibliotecznych, muszą być tak dobrane, aby występowała całkowita zgodność deklaracji zapowiadającej prototyp i definicji funkcji. Po spełnieniu takiego wymagania można zrezygnować z użycia zbioru nagłówkowego, ale tylko wtedy, gdy jego włączenie dyrektywą #include nie jest wymagane z punktu widzenia innych elementów programu. elementów programu.

Przykład

```
main()
{
  int printf(char *, ...);
  printf("Hello, I am JanB");
}
```

Mimo stwierdzenia, że włączanie zbiorów nagłówkowych nie jest wymagane, należy mieć na uwadze, że umiejętne posługiwanie się nimi może istotnie skrócić zapis programu źródłowego. Z tego względu, każdy poważny użytkownik systemu Turbo C powinien poświęcić nieco czasu na zapoznanie się z zawartością tych zbiorów. A oto krótki ich przegląd:

ALLOC.H — zawiera deklaracje funkcji do zarządzania pamięcią operacyjną;

ASSERT.H — zawiera makrodefinicję orzekającą o spełnieniu zadanej relacji;

BIOS.H — zawiera deklaracje funkcji odwołujących się do systemu BIOS;

CONIO.H — zawiera deklaracje funkcji umożliwiających współpracę z konsolą;

CTYPE.H — zawiera informacje wykorzystywane przez makrodefinicje do klasyfikowania znaków;

DIR.H — zawiera deklaracje i definicje umożliwiające posługiwanie się katalogami;

DOS.H — zawiera makrodefinicje i deklaracje umożliwiające wywoływanie funkcji systemu MS-DOS;

ERRNO.H — zawiera definicje umożliwiające posługiwanie się mnemonikami kodów błędów;
FCNTL.H — zawiera definicje wykorzystywane przez funkcję biblioteczną do otwierania plików;

FLOAT.H — zawiera makrodefinicje wykorzystywane przez funkcje biblioteczne do wykonywania operacji zmiennopozycyjnych;

IO.H — zawiera deklaracje wykorzystywane przez funkcje biblioteczne do wykonywania operacji wejścia-wyjścia niskiego poziomu;

LIMITS.H — zawiera makrodefinicje dostarczające informacji o parametrach implementacji;

MATH.H — zawiera deklaracje prototypowe funkcji matematycznych oraz makrodefinicję danej zmiennopozycyjnej „nieskończenie dużej";

MEM.H — zawiera deklaracje funkcji dotyczących pamięci operacyjnej;

PROCESS.H — zawiera deklaracje wykorzystywane przez funkcje do tworzenia procesów;

SETJMP.H — zawiera deklarację typu i deklaracje funkcji wykorzystywanych przez funkcje do zapamiętywania i odtwarzania środowiska wykonywania programu;

SHARE.H — zawiera makrodefinicje wykorzystywane przez funkcje realizujące współdzielenie zbiorów;

SIGNAL.H — zawiera makrodefinicje wykorzystywane przez funkcje do definiowania i obsługiwania sygnałów.
STDARG.H — zawiera makrodefinicje ułatwiające uzyskanie dostępu do argumentów funkcji ze zmienną liczbą argumentów;

STDDEF.H — zawiera podstawowe deklaracje typów i makrodefinicje;

STDIO.H — zawiera deklaracje typów i makrodefinicje wykorzystywane przez standardowe funkcje wejścia-wyjścia;

STDLIB.H — zawiera deklaracje często używanych funkcji;
STRING.H — zawiera deklaracje funkcji do przetwarzania ciągów znaków;

SYS/STAT.H — zawiera makrodefinicje wykorzystywane podczas otwierania plików i tworzenia zbiorów;

TIME.H — zawiera deklaracje struktur i deklaracje funkcji do udostępniania i przetwarzania danych określających porę dnia;

VALUES.H — zawiera makrodefinicje określające parametry komputera.

Procedury INTR i MSDOS w MS-Pascalu

dokończenie ze s. 14

Użycie procedur INTR oraz MSDOS ma jednak następujące zalety:

- rozszerza pole zastosowań języka MS-Pascal,
- ułatwia tworzenie programów użytkowych o bezkolizyjnym współdziałaniu z systemem DOS,
- umożliwia wykorzystanie bibliotek funkcji DOS utworzonych dla Turbo Pascala.

LITERATURA

[1] Bielecki J.: Turbo Pascal. Informatyka, nr 7, 1987
[2] Borland Intnl: Turbo Pascal Version 3.0. Reference Manual, 1985
[3] IBM Personal Computer Disk Operating System. Technical Reference, Update Number 1, March 1986
[4] Lauer A., Wallwitz B.: Turbo-Machine Version 1.00, Wiesbaden, 1985
[5] Microsoft Corp.: Microsoft Macro Assembler. Reference Manual for the MS-DOS Operating System. Document Number 8451-300-00 (Appendix C), Bellevue (WA), 1984
[6] Microsoft Corp.: Microsoft Pascal. Reference Manual. Document Number 8501-330-08, Bellevue (WA), 1985
[7] Microsoft Corp.: Microsoft Pascal Compiler for MS-DOS Operating System. User's Guide. Document Number 8511L-330-05, Bellevue (WA), 1985
[8] Raznowiecki A., Syfert A.: Struktura systemu operacyjnego PC-DOS, cz. 3—6. Informatyka, nr 11—12, 1987, nr 1—3, 1988
[9] Völzing P.: MS DOS im Detail IWT-Verlag München, 1985.

Z przykrością informujemy Czytelników, że począwszy od numeru lipcowego br. cena INFORMATYKI została ustalona na 250 zł za egzemplarz.

Nowa cena obowiązuje nowych prenumeratorów oraz sprzedaż odręczną. Jednak nie wyklucza się ewentualnej dopłaty różnicy (pomiędzy nową a starą ceną) do już opłaconej prenumeraty na drugie półrocze br. O ewentualnej dopłacie prenumeratorzy zostaną powiadomieni pisemnie przez Dział Kolportażu Wydawnictwa SIGMA.

Dotychczasowa cena ulgowa — 50 zł — zostaje utrzymana w prenumeracie do końca 1988 roku.

Jednocześnie informujemy, że od stycznia 1989 roku cena INFORMATYKI będzie wynosić:

300 zł za egzemplarz — cena normalna
60 zł za egzemplarz — cena ulgowa.

Szczegóły podano w „Warunkach prenumeraty" na s. 11.

HENRYK BIESIADA
JERZY GRZYWOCZ
Instytut Informatyki
Politechnika Śląska

Komputerowe przetwarzanie wyników badań medycznych

Pracownicy naukowi medycyny często zgłaszają zapotrzebowanie na zestawy programów związanych z opracowaniem wyników eksperymentów. Zaspokojenie tego zapotrzebowania pochłania wiele czasu programistów oraz znacznie obciąża sprzęt informatyczny, ponieważ na ogół każde zamówienie na odpowiednie programy jest realizowane indywidualnie. Powstaje w ten sposób wiele podobnych, lecz mało uniwersalnych programów, stosowanych najczęściej jednorazowo.

Wymagania wynikające ze specyfiki badań medycznych można sprecyzować w kilku punktach:

- powstały system programów musi stanowić jednolitą całość, możliwą do zaimplementowania na przeciętnie rozbudowanej, dostępnej w danym ośrodku konfiguracji sprzętowej,
- konieczna jest maksymalna prostota obsługi całego systemu, aby również nieinformatycy mogli z nim w miarę swobodnie współpracować;
- przechowywane dane muszą być chronione przed niepowołanym dostępem (m.in. do zachowania tajemnicy lekarskiej),
- struktury danych muszą być elastyczne (tzn. zdolne do przechowania wyników różnego rodzaju badań w postaci znaków lub liczb), proste do interpretacji (odpowiadające logicznie najczęściej spotykanym w codziennej praktyce formom reprezentacji informacji, np. tabelom) i odporne na często pojawiające się w wynikach badań medycznych niejednorodności (np. pojedyncze specyficzne przypadki w ciągu przypadków typowych),
- system musi mieć zbiór programów do opracowania danych (np. metodami statystycznymi) oraz mieć łatwość dołączania podprogramów ukierunkowanych na potrzeby poszczególnych użytkowników.

KONCEPCJA SYSTEMU PROGRAMÓW PRZETWARZAJĄCYCH DANE MEDYCZNE

Po analizie wymagań stawianych systemowi należy przyjąć, że oprogramowanie takie musi realizować następujące funkcje:

— zakładania, modyfikowania, podporządkowania i raportowania zbioru z danymi,
— ochrony zbioru przed niepowołanym dostępem,
— opracowania danych,
— specjalizowanego przetwarzania danych.

Równocześnie powinno ono mieć takie cechy, jak modularność, jednolitość środków komunikacji modułów z użytkownikiem, prostota obsługi, ochrona przed skutkami błędów użytkownika.

Można więc dokonać próby podziału całego oprogramowania na kilka modułów, realizujących poszczególne funkcje i zapewniających przy tym całej strukturze wymagane cechy.

Aby uniknąć wielokrotnego pisania fragmentów programu o podobnych funkcjach i zapewnić efektywne współdziałanie modułów, celowe jest stworzenie z tych modułów struktury kilkupoziomowej, w której moduły poziomu wyższego wykorzystują moduły znajdujące się na poziomie niższym. Na pierwszym, najniższym poziomie, powinny się znajdować procedury bezpośrednio działające na zbiorze danych, np. procedury odczytywania czy zapisywania informacji na fizyczny nośnik.

Drugi, wyższy poziom jest przeznaczony dla modułów realizujących wymienione funkcje. Pierwsze dwie funkcje można rozdzielić między grupę ściśle ze sobą współdziałających modułów, takich jak:

— definiujący strukturę danych,
— zapisujący i modyfikujący dane,
— porządkujący dane,
— sporządzający raporty.

Oddzielny moduł stanowi zestaw programów opracowujących dane. Ostatnią funkcję można realizować przy użyciu modułu będącego zbiorem podprogramów pisanych dla potrzeb lekarzy prowadzących swoje badania naukowe. Moduł ten można potraktować jako swoisty „bank metod", do którego są dodawane procedury potrzebne kolejnym użytkownikom.

Do zapewnienia całemu systemowi jednolitości, prostoty i bezpieczeństwa danych konieczne jest wprowadzenie (na najwyższym — trzecim poziomie hierarchii modułów) modułu integrującego całą strukturę, zarządzającego pozostałymi modułami i prowadzącego dialog z użytkownikiem.

Przy tworzeniu systemu należy rozwiązać kilka problemów związanych z dialogiem między użytkownikiem a programem, logiczną strukturą zbioru z danymi, sposobem przetwarzania danych oraz sposobem prezentacji przechowywanych danych oraz wyników ich przetwarzania.

Problemy dialogu użytkownik-system

Jednym z ważniejszych warunków funkcjonalności rozważanego oprogramowania jest możliwość jego obsługi przez użytkownika. Dialog z programem powinien się więc charakteryzować:

— jednoznacznymi i czytelnymi komunikatami o zdarzeniach programowych,
— prostotą środków do sterowania przez użytkownika pracą programu,
— istnieniem systemu „przepowiedzi" pomagających użytkownikowi w wyborze właściwego wariantu działania,
— dobrą ochroną przed skutkami akcji użytkownika niezgodnej z syntaktycznymi zasadami dialogu (np. przed skutkami wysłania do procesora znaku nie należącego do gramatyki definiującej dialog),
— istnieniem systemu ostrzegania przed skutkami akcji poprawnych z punktu widzenia działania programu, ale potencjalnie groźnych dla użytkownika przez swą radykalność (np. usuwanie zbioru z danymi).

Proponuje się dialog kilkupoziomowy. Na pierwszym poziomie byłby określony rodzaj akcji na zbiorze danych lub jego elementach (tzw. menu), na dalszych poziomach natomiast, na które przechodziłoby się po wyborze określonej pozycji spośród przedstawionych w wykazie na poziomie pierwszym, następowałoby sprecyzowanie żądania użytkownika.

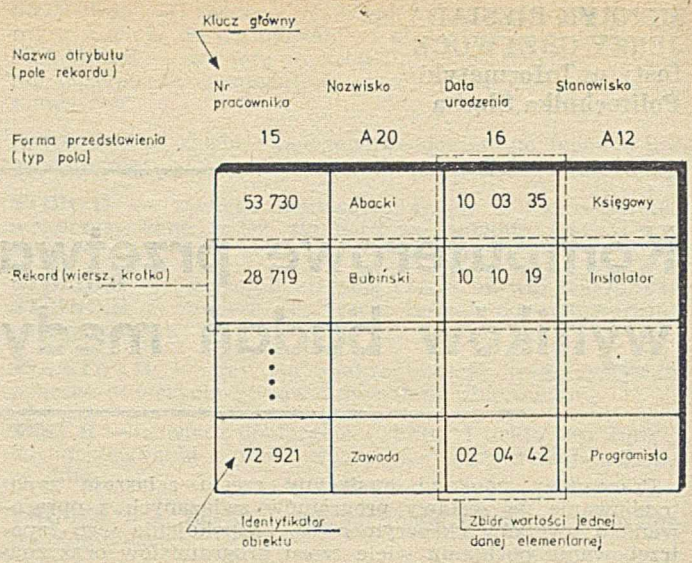
Logiczna struktura zbioru z danymi

Struktury danych pochodzących z badań medycznych w przeważającej większości nie są zbyt złożone w sensie wzajemnych powiązań, często natomiast mają bardzo wiele elementów. Przykładem może być struktura danych pochodząca z badań nad diagnostyką i leczeniem wnętrostwa, przedstawiona na rys. 1.

Struktura z rys. 1 jest prostym drzewem, w którym węzłami są rekordy. Drzewo to jest pewnym uproszczeniem struktury, powstałej po pierwszej rozmowie z lekarzem prowadzącym badania. Schemat ten nie może jeszcze stanowić podstawy do planowania struktury zbioru danych. Jest on bowiem przypadkiem szczególnym, jednym z wielu możliwych. A przecież funkcją systemu jest jego uniwersalność, którą może zapewnić relacyjna struktura danych (rys. 2).

Okazuje się, że każdą strukturę danych, również hierarchiczne struktury proponowane przez lekarzy, można przedstawić w postaci relacji. W celu zmniejszenia redundancji danych relację dalej się przekształca, co jest nazywane procesem normalizacji. Jako ostateczny wynik normalizacji przyjmuje się najczęściej zbiór relacji w trzeciej postaci normalnej.

Przy dużym rozgałęzieniu drzew i wielu różnych grupach powtórzeń cała struktura drzewiasta rozkłada się przy przechodzeniu na znormalizowaną strukturę relacyjną na wiele relacji, co może czasem utrudniać ogarnięcie struktury danych.



Rys. 2. Przykład opisujący podstawowe pojęcia relacyjnej struktury danych (pojedyncze pole tabeli zawiera wartość atrybutu)

Opracowanie danych

Rozpatrywane zagadnienie obejmuje różnorodne metody przetwarzania. Najczęściej wymagane jest statystyczne opracowanie danych, z użyciem programów pisanych na zamówienie poszczególnych użytkowników lub specjalnych pakietów programów statystycznych. Pakiet programów statystycznych powinien umożliwiać:

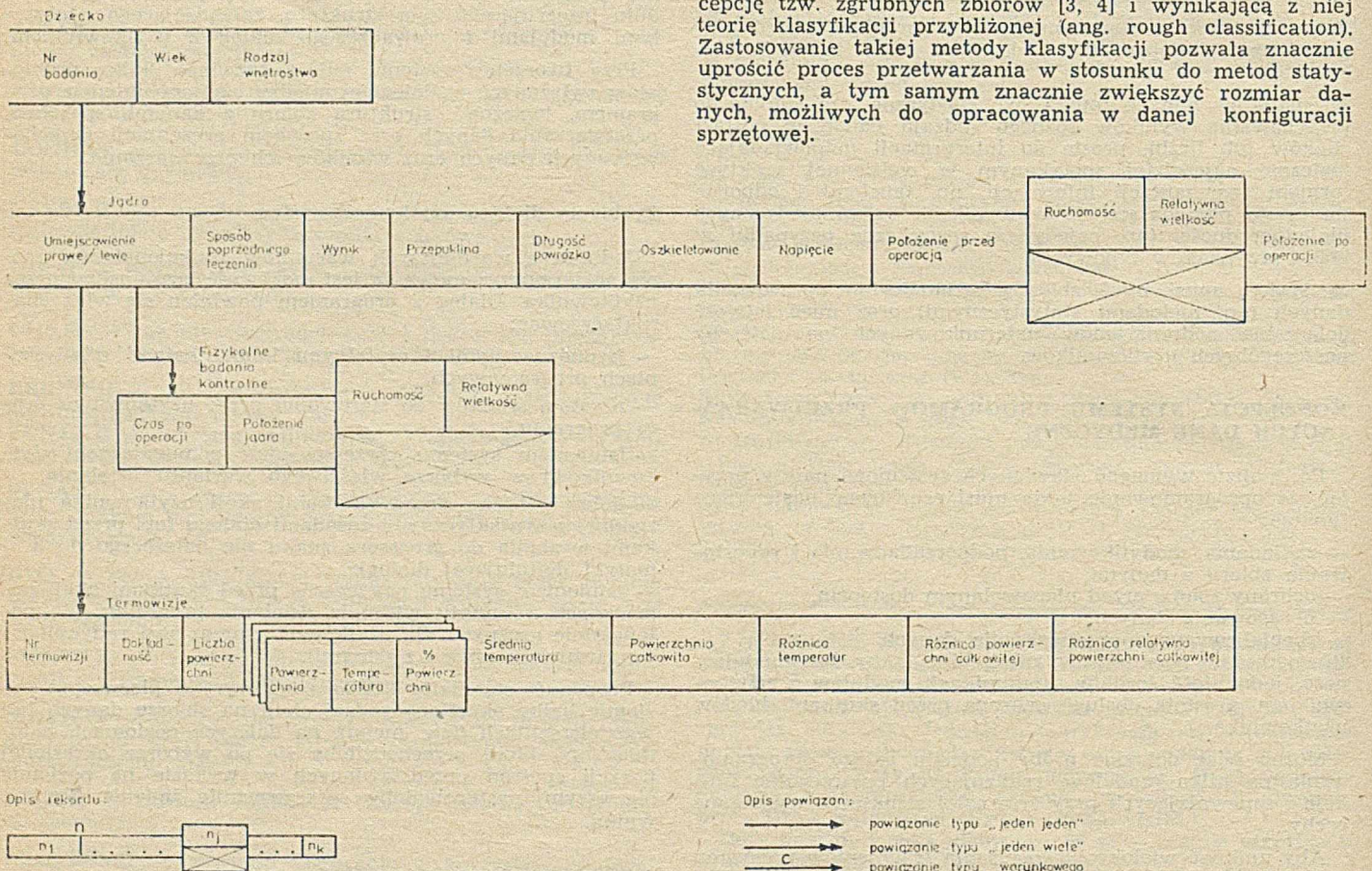
— wyznaczenie podstawowych charakterystyk danych, takich jak: średnia (ważona, arytmetyczna), przedział ufności, wariancja, odchylenie standardowe,

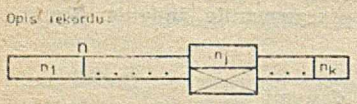
— przeprowadzenie obliczeń związanych z analizą regresji, wariancji oraz analizą czynnikową.

Oczywiście analiza statystyczna nie jest jedynym narzędziem do opracowania danych. Jako przykład można rozważyć następujące zagadnienie.

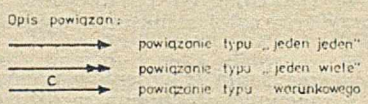
Dany jest zbiór obiektów podzielony na pewne podzbiory (klasy). Wszystkie obiekty są opisane pewną skończoną liczbą atrybutów (cech), przybierających wartości z określonych dziedzin. Należy określić, które atrybuty i w jakim stopniu wpływają na dokonany podział.

Można to zagadnienie rozwiązać posługując się metodami innymi niż statystyka matematyczna, np. stosując koncepcję tzw. zgrubnych zbiorów [3, 4] i wynikającą z niej teorię klasyfikacji przybliżonej (ang. rough classification). Zastosowanie takiej metody klasyfikacji pozwala znacznie uprościć proces przetwarzania w stosunku do metod statystycznych, a tym samym znacznie zwiększyć rozmiar danych, możliwych do opracowania w danej konfiguracji sprzętowej.





n — nazwa rekordu
n_1 — identyfikator rekordu
n_j — nazwa danej elementarnej, pole rekordu
n_j — pole n_j jest nie wypełnione w zależności od pewnych warunków



Rys. 1. Przykład nieznormalizowanej struktury danych

Sporządzanie raportów

System korzysta ze środków raportowania zdefiniowanych przez jego elementy tj. zastosowaną bazę danych lub pakiet programów statycznych. Często jednak konieczna jest pewna mniej typowa forma wydruku, taka jak nieznormalizowana tabela, diagram lub wykres. Podprogramy sprządzające takie raporty powinny być umieszczane w module realizującym funkcję specjalizowanego przetwarzania danych.

OPIS IMPLEMENTACJI

Ogólną strukturę wielomodułowego systemu analizy danych (zwanego dalej WSAD) przedstawiono na rys. 3, w postaci diagramu przepływu danych. Na rysunku są widoczne główne elementy projektowanego systemu. Pierwszym elementem jest MODUŁ_ZARZĄDZAJĄCY, który zawiera:

1) makropolecenie **WSAD**
— rozpoczynające i kończące pracę całego systemu WSAD,
— opisujące system na żądanie użytkownika,
— umożliwiające uruchomienie pozostałych modułów;

2) zespół makropoleceń **STATYSTYKA**
— zapewniający wybór jednego z rodzajów analizy statystycznej,
— opisujący na życzenie użytkownika wybrany rodzaj analizy,
— uruchamiający program DAST,
— zapewniający gospodarkę zbiorami danych dla programu analizy statystycznej;

3) zespół makropoleceń **KLASYFIKACJA**
— zapewniający uruchomienie systemu przybliżonej analizy danych (SPAD),
— opisujący na życzenie użytkownika możliwości systemu SPAD,
— uruchamiający program DAKL.

Drugim elementem systemu jest MODUŁ_BAZY_DANYCH. Jego zasadniczą częścią jest relacyjna baza danych DBMS. W skład modułu wchodzą dwa niezależne programy i obsługujące je makropolecenia:

1) program o nazwie **PROGRAM DBMS**, zarządzający zbiorami relacyjnej bazy danych; jest to interpreter języka manipulowania relacjami;

2) makropolecenie **DBMS**, obsługujące pracę programu DBMS (przez obsługę pracy programu rozumie się jego ładowanie do pamięci operacyjnej, przydział urządzeń zewnętrznych i czasu procesora, zainicjowanie programu oraz obsługa zdarzeń programowych);

3) program **KSAD**, umożliwiający użytkownikowi konwersacyjną aktualizację danych zawartych w zbiorach relacyjnej bazy danych;

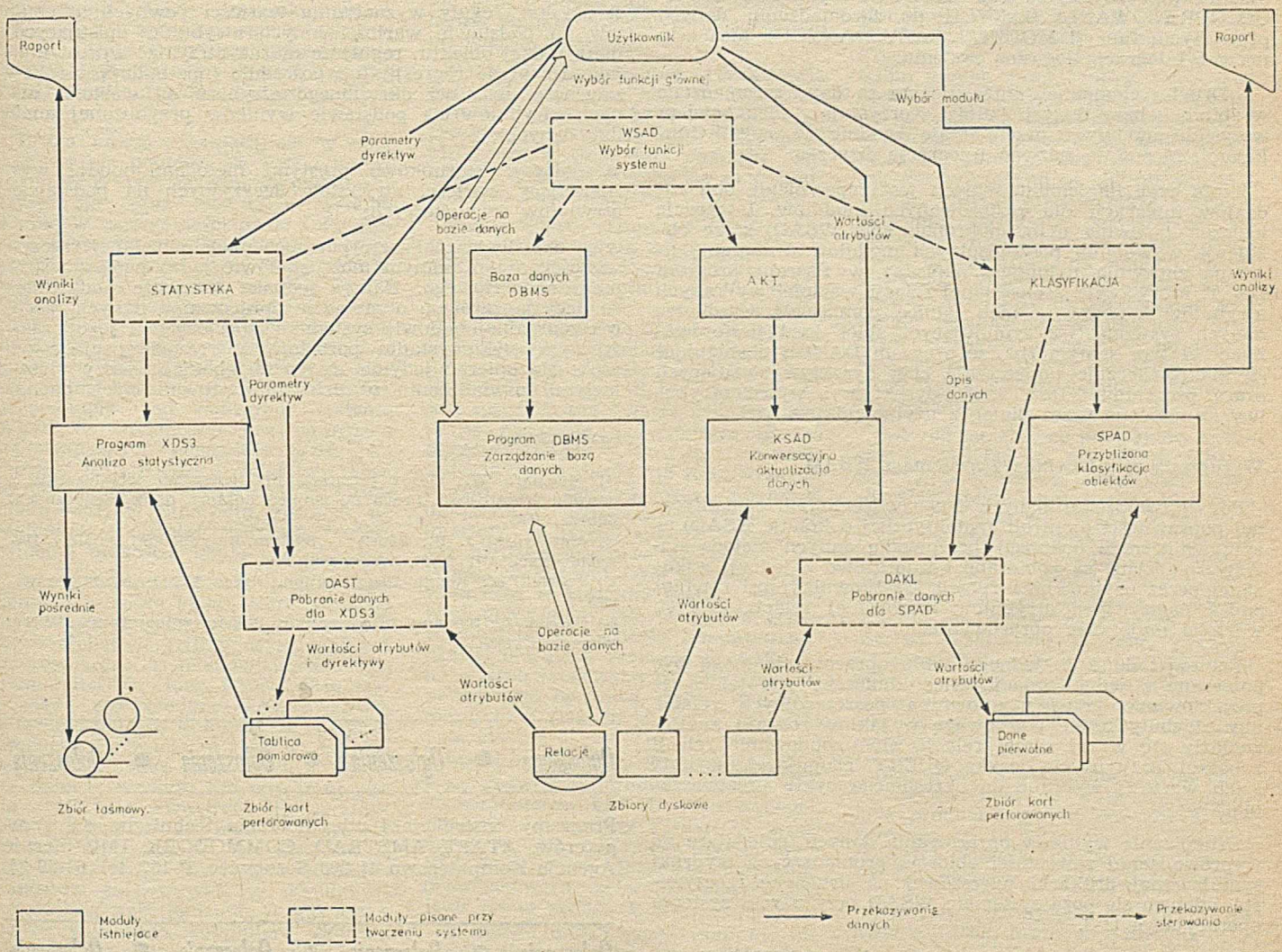
4) makropolecenie **AKT** obsługujące pracę programu KSAD.

Trzecim elementem systemu jest MODUŁ_OPRACOWANIA_DANYCH, złożony z dwóch niezależnych od siebie modułów: MODUŁU_OPERACJI_STATYSTYCZNYCH i MODUŁU_PRZYBLIŻONEJ_ANALIZY_DANYCH. MODUŁ_OPERACJI_STATYSTYCZNYCH zawiera następujące programy:

1) program **XDS3**, realizujący analizę statystyczną,
2) program **DAST**, pobierający określone dane ze zbiorów bazy danych DBMS i podający je na wejście programu XDS3.

MODUŁ_PRZYBLIŻONEJ_ANALIZY_DANYCH obejmuje:

1) zbiór programów i makropoleceń **SPAD** (system przybliżonej analizy danych), przeznaczony do rozwiązywania zagadnień zgrubnej klasyfikacji obiektów,



Rys. 3. Implementacja systemu przetwarzania danych medycznych, diagram przepływu danych

2) program **DAKL**, przekazujący dane ze zbiorów bazy danych do zbioru z danymi wejściowymi dla systemu SPAD

Czwartym elementem systemu, nie uwidocznionym na rys. 3 jest moduł specjalizowanego przetwarzania danych. Przewiduje się, że będzie to jeden lub kilka programów w Pascalu, wykonujących zadane przez użytkownika operacje na danych i zapisujących wyniki tego przetwarzania do innych lub tych samych relacji albo też raportujących te wyniki w zadany sposób. Konkretną postać tego modułu określają dopiero pierwsi użytkownicy systemu WSAD.

Realizacja systemu WSAD

Główne funkcje systemu WSAD są realizowane przez istniejące już i wielokrotnie wypróbkowane środki programowe systemu GEORGE-3 (ODRA 1305). System WSAD zawiera relacyjną bazę danych, konwersacyjny system aktualizacji danych KSAD i pakiet programów analizy statystycznej XDS3.

W swej obecnej postaci system WSAD nie umożliwia dokonania klasyfikacji przybliżonej. Funkcję tą mógłby spełniać dopiero po dołączeniu do niego zbioru programów i makropoleceń umożliwiających jej wykonanie (np. systemu SPAD [2]).

Współpraca modułów systemu WSAD

Omówione moduły systemu WSAD, tak bardzo różne w swoich funkcjach i budowie, są integrowane przez dwa elementy. Pierwszym z nich jest pewne środowisko programowe w postaci makropoleceń tworzących MODUŁ-ZARZĄDZAJĄCY. Makropolecenia te sprawiają, że uruchomienie funkcji każdego z modułów jest (z punktu widzenia użytkownika) odwołaniem się do pewnych części większej całości. Ponadto użycie funkcji jednego modułu wydaje się być logicznym następstwem użycia funkcji innego modułu (np. użycie funkcji MODUŁU-STATYSTYCZNEGO-OPRACOWANIA-DANYCH po zgromadzeniu danych przy użyciu funkcji MODUŁU-BAZY-DANYCH). Można to nazwać integracją logiczną systemu.

Drugim elementem integrującym są dane zgromadzone w zbiorach bazy danych DBMS. Korzystają z nich bowiem wszystkie moduły i przetwarzają zgodnie ze swymi funkcjami. Można to nazwać integracją fizyczną.

Istotą prac implementacyjnych jest zapewnienie obu rodzajów integracji dla zastosowanych modułów. Integrację logiczną zapewnia m.in. makropolecenie WSAD, które obsługuje wywołania poszczególnych modułów. Integrację fizyczną zapewnia natomiast napisany w Pascalu program DAST, który czyta z określonej relacji wartości wybranych atrybutów i tworzy z nich tablicę pomiarową w zbiorze danych systemu G-3 symulującym zbiór kart perforowanych. W tym samym zbiorze program DAST i obsługujące go makropolecenie umieszczają ciąg dyrektyw sterujących pracą programu XDS3. Treść dyrektyw i wybór atrybutów są ustalone przez dialog z użytkownikiem.

Współpraca użytkownika z systemem WSAD

Po wprowadzeniu danych i sprawdzeniu ich semantycznej poprawności (kontroli syntaktycznej dokonuje KSAD) użytkownik może dokonać opracowania danych metodą statystyczną. Dane są pobierane każdorazowo z jednej, wskazanej przez użytkownika relacji. Użytkownik musi również podać nazwy tych atrybutów rozważanej relacji, których wartości mają ulec opracowaniu.

Ponieważ dane do jednego cyklu opracowania mogą być pobierane z jednej relacji, więc konieczne jest takie zaprojektowanie przez użytkownika poszczególnych relacji, aby atrybuty brane pod uwagę w jakimś rodzaju analizy znalazły się w tej samej relacji, albo zbudowanie relacji roboczej, przy użyciu języka QUERY, grupującej atrybuty i ich wartości wybrane do określonego cyklu przetwarzania.

Drukowanie wyników opracowania danych, polegające na wyprowadzeniu zawartości zbiorów gromadzących wydruki symulowanej drukarki wierszowej na drukarkę rzeczywistą, odbywa się poza systemem WSAD przy użyciu systemu operacyjnego G-3.

Ponieważ system zarządzania bazą danych DBMS nie ma wbudowanych środków chroniących dane w poszczególnych relacjach, więc jedynym na razie sposobem zabezpieczenia danych jest zastosowanie mechanizmów systemu G-3.

Użytkownik współpracuje z systemem WSAD za pośrednictwem trójpoziomowego dialogu. Na pierwszym poziomie użytkownik ma przedstawione następujące możliwości (menu):

0 — koniec pracy w systemie WSAD (powrót do G-3)
1 — operacje na bazie danych
2 — konwersacyjna aktualizacja danych
3 — analiza statystyczna
4 — klasyfikacja przybliżona
? — informacja ogólna

System WSAD jako zalążek systemu ekspertowego

Jednym z podejść do tworzenia systemu ekspertowego jest podejście indukcyjne, polegające na obserwacji przypadków szczególnych, następnie dokonaniu uogólnień (czyli budowaniu pewnego modelu odwzorowującego rozważane przypadki szczególne) i wreszcie implementacji tego modelu. Ideę takiego modelu określa zapis:

if ⟨warunki⟩ then ⟨akcja⟩

przy czym przez warunki można rozumieć dane wartości pewnego podzbioru atrybutów opisujących obiekt (zwanych atrybutami warunkowymi lub obserwowalnymi), a przez akcję można rozumieć podanie wartości innego podzbioru atrybutów opisujących obiekt (zwanych atrybutami decyzyjnymi) na podstawie wartości atrybutów warunkowych i reguł zakodowanych w modelu (np. równań regresji lub tablicy decyzyjnej).

Ponieważ system WSAD pozwala:

- zgromadzić pewną liczbę danych o przypadkach szczególnych;
- uzyskać reguły wyznaczania wartości pewnych atrybutów na podstawie wartości innych atrybutów opisujących dany rodzaj obiektu; reguły te można otrzymać wprost (np. jako równanie regresji) lub pośrednio (np. budując tablicę zależności wartości określonego atrybutu od wartości innych atrybutów na podstawie wyników przybliżonej analizy danych);
- dołączyć stosunkowo prostymi metodami moduł wyznaczający wartości atrybutów decyzyjnych na podstawie atrybutów warunkowych;

więc umożliwia realizację pewnego modelu badanej rzeczywistości, a tym samym może stanowić jądro pewnego systemu ekspertowego. Należy jednak stanowczo podkreślić, że decyzja zarówno o metodzie opracowania danych, jak i o ewentualnej budowie systemu ekspertowego i jego praktycznym wykorzystaniu pozostaje w rękach specjalisty i może się opierać jedynie na jego głębokiej wiedzy i rozległym doświadczeniu w dziedzinie prowadzonych badań.

LITERATURA

[1] Madziara W.: Obraz termowizyjny jąder po ich leczeniu z powodu wnętrostwa. Praca na stopień doktora nauk medycznych, Zabrze, 1985
[2] Marczyński J. B.: SPAD — System Przybliżonej Analizy Danych. Prace IPI PAN, Warszawa, 1985
[3] Pawlak Z.: Rough Classification. Prace IPI PAN, Warszawa, 1985
[4] Pawlak Z.: Systemy informacyjne. Podstawy teoretyczne. WNT, Warszawa, 1984.

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

Programy, instrukcje i udoskonalenia techniczne dla komputerów, ATARI, AMSTRAD, COMMODORE, IMB oferuje Agencja Komputerowa 41-200 Sosnowiec, P-157, tel. 63-29-35.
EO/423/87

Ogłoszenia • Ogłoszenia • Ogłoszenia • Ogłoszenia

INFOSYSTEM '88

Po niejako próbnym starcie we Wrocławiu w 1987 roku, od bieżącego roku Międzynarodowe Targi Elektroniki, Telekomunikacji i Techniki Komputerowej **Infosystem** weszły do poznańskiego kalendarza targowego. Odbywały się na terenie targowym w dniach od 25 do 29 kwietnia, a ekspozycja ok. 200 wystawców zajmowała cztery duże i kilka małych pawilonów.

Nazwa targów jest bardzo myląca. Elektronika użytkowa i pomiarowa oraz telekomunikacja zajmowały w sumie poniżej 10% ekspozycji i ginęły w masie komputerów, a dokładniej — mikrokomputerów. Większość stoisk była zastawiona sprzętem klasy IBM PC, różnej mocy i proweniencji. W stosunku do ubiegłego roku jedyną zmianą było pojawienie się wielu komputerów klasy Super AT (z procesorem 386). Biorąc pod uwagę jakość rozwiązań architektonicznych, można liczyć na to, że za kilka lat ten właśnie model ma szanse stać się w miarę stabilnym standardem w grupie mikrokomputerów osobistych. Kilka firm pokazywało także mikrokomputery PS/2, ale bez specyficznego oprogramowania. Firma Data Star prezentowała nawet Model 80 bez pokrywy, tak aby każdy mógł sprawdzić, co jest w środku.

Drugą rzeczą rzucającą się w oczy jest rosnąca popularność sieci i wielodostępu. W tym drugim wypadku korzysta się coraz częściej z systemu Xenix. W szczególności firma CSK oferuje takie rozwiązania, wraz z oprogramowaniem sieciowym MicroLAN-X, które po raz pierwszy pozwala łączyć komputery pracujące pod nadzorem PS-DOS-u i Xenixa. Jeśli mowa o tej firmie, to jej ciekawym produktem jest komputer Quatro 386, z czterema niezależnymi kartami grafiki kolorowej, pozwalający pod nadzorem systemu WDOS na wielodostęp bez ograniczeń wprowadzanych normalnie przez terminale. Jeśli chodzi o właściwe sieci, to oferowana jest sieć Arcnet oraz rozmaite mutacje Ethernetu, różniące się szybkością transmisji, rodzajem i maksymalną długością kabla.

Na dwóch stoiskach pokazywano komputery spoza klasy mikro. Spośród wystawców z państw socjalistycznych tylko **Robotron** (NRD) wystawiał coś interesującego, a mianowicie superminikomputer 32-bitowy RVS K 1840, stanowiący odpowiednik VAX-a 11/780. Wiadomo było od kilku lat, że rozmaici producenci przymierzają się do tej maszyny, ale dopiero teraz był to faktyczny debiut targowy. Niestety, rzut oka na urządzenia zewnętrzne (taśmy i dyski) przekonuje, że trudno liczyć na niezawodną pracę — ten sam sprzęt był stosowany w odpowiednikach PDP-11 i zachowywał się skandalicznie. Należy mieć nadzieję, że gdy tylko nabywcy takiego komputera skończy się gwarancja, to będzie mógł natychmiast zakupić w firmie rzemieślniczej, spółdzielczej lub polonijnej odpowiednio niezawodne dyski Winchester. Komputer jest wyposażony w system operacyjny SVP 1800 (odpowiednik VMS) lub w MUTOS 1800 (kopia Unixa). Ponieważ procesor oferuje tryb emulacji instrukcji komputera PDP-11, a system VMS definiuje dla programów wykonywanych w tym trybie środowisko systemu RSX-11, to komputer RVS stanowi drogę migracji oprogramowania działającego na komputerach SM-4.

Ze strony polskiej jedynym wystawcą mikrokomputera była spółdzielnia **Comtech**, znana z bardzo ciekawych rozwiązań w dziedzinie rewitalizacji komputerów SM-4. Tym razem wystawiono własny produkt, będący rozbudowanym odpowiednikiem komputera PDP-11/24. Wykorzystano radzieckie procesory serii 1300, własne rozwiązanie rozszerzenia pamięci do 4 MB, podsystem pamięci dyskowej typu Winchester (w sumie do 760 MB) i dyskietek 5,25-calowych lub 3,5-calowych, a także dysk wirtualny do 8 MB w dodatkowej pamięci półprzewodnikowej. Oczywiście, można dołączać standardowe terminale, drukarki itp. Komputer pracuje pod nadzorem systemu RSX-11M, a będzie dostępny w sprzedaży w IV kwartale 1988 roku. Cena jeszcze nie była ustalona, ale zapowiadano, że będzie konkurencyjna w stosunku do prostszych rozwiązań.

Również firma **Danpol** prezentowała duże, wielodostępne konfiguracje komputerowe serii DP 2000, z 32-bitowym procesorem, pamięcią operacyjną do 32 MB i odpowiednio dużą pamięcią dyskową. Terminale dołącza się za pośrednictwem inteligentnego sterownika.

Firma **Elwro**, gigant polskiego przemysłu, prezentowała szeroką ofertę, z mikrokomputerem Elwro 801 klasy AT na czele. Według oficjalnej informacji komputer ma polską obudowę i płytę główną (rzeczywiście układy małej skali integracji są produkcji Cemi, choć inne oczywiście pochodzą z Dalekiego Wschodu). Reszta, w tym klawiatury, jest sprowadzana z zagranicy. Być może to pozwoliło na ustalenie bardzo atrakcyjnej ceny — 3,5 mln zł za komputer AT z pamięcią 512 KB i dyskiem stałym. Na dodatek cena obejmuje oficjalnie licencjonowany MS-DOS (niestety po angielsku), MS Basic (to już bardziej wątpliwa atrakcja) i MS Works. W każdym razie stanowi to pewien krok w kierunku legalnej dystrybucji oprogramowania. W stoisku Elwro wystawiała także firma **Mannesmann-Tally**, oferująca rozmaite drukarki — od małych biurowych po specjalizowane drukarki bankowe, szybkie drukarki mozaikowe (400 znaków/s) i drukarki laserowe.

Instytut Systemów Sterowania z Katowic rozwija serię odpowiedników mikrokomputerów LSI-11. Obecny model, Mera 680, jest klasy PDP-11/23, z pamięcią 4 MB, systemem operacyjnym RSX 60M i może służyć do sterowania przemysłowego, zbierania danych lub jako stacja graficzna.

Z zagranicznych potentatów nie zawiodły firmy **Hewlett-Packard** (system HP 3000, drukarki laserowe) oraz **Wang** (wielostanowiskowe systemy CS+DS, zgodne programowo z poprzednimi, znanymi w Polsce komputerami). Zdziwił mnie brak przedstawicielstwa **ICL**, po raz pierwszy na imprezie tego typu.

Było trochę drobnych ciekawostek. Firma **Mera-Błonie** prezentowała podajnik kartek papieru do drukarki D-100. Biorąc pod uwagę problemy z korzystaniem przez tę drukarkę z perforowanej składanki, podajnik powinien znakomicie podnieść komfort pracy. Firma **Mera Poltik**, znana z produkcji budzików, wystawiła bardzo ładny, mały i zwarty ploter formatu A4. Jeśli zapowiadana (niska) cena i rozmiar produkcji okażą się realne, to będzie to doskonałe wyposażenie osobistych komputerów, pozwalające na kreślenie niewielkich rysunków nie wymagających idealnej precyzji. Ploter jest programowo zgodny z urządzeniami Rolanda.

Bardzo wiele firm prezentowało małe systemy wydawnicze (ang. desktop publishing). Drukarki laserowe i analizatory obrazu stają się już bardzo popularne. Firma **Rank Xerox** nadal niestety nie oferuje polskiej wersji programu Ventura, nie są także ustalone zasady ewentualnego licencjonowania tej wersji innym producentom. **Dom Handlowy Nauki** oferuje do swoich drukarek laserowych polskie litery, akceptowane również przez Venturę. Kilku innych producentów zapowiada podobne produkty w bliżej nie określonej przyszłości. **Firma Cyfronex**, która poprzednio zajmowała się profesjonalnymi programami obsługi fotoskładarek korzystającymi ze specjalnych klawiatur, obecnie oferuje także oprogramowanie Cyfroset, pozwalające na składanie testów (także matematycznych) drukowanych na drukarkach laserowych. Program jest niesłychanie uniwersalny, pozwala na łączenie materiału tekstowego i graficznego. Specyfika polskich warunków powoduje, że rozważa się nawet składanie za pomocą tego programu wielonakładowych tygodników.

Spośród mniejszych programów szczególnie spodobał mi się edytor tekstowy TAG, produkcji spółdzielni **Info-Service.** Edytor jest bardzo prosty w użyciu, pozwala wybierać z kilku krojów i wielkości znaków, ma polskie litery, a także cyrylicę. Szczególne wrażenie robi szybkość pracy — zupełnie nie odczuwa się tego, że edytor pracuje w trybie graficznym. Przemyślne upakowanie słownika pozwala na sprawdzanie pisowni bez zbytniego narzutu czasowego i pamięciowego. Mam nadzieję, że będziemy mogli wkrótce przedstawić szerszy opis tego programu.

Największym urządzeniem była chyba naświetlarka do klisz używanych przy produkcji płyt drukowanych; dokładność do 10 mikrometrów, rozmiary dużej drukarki wierszowej, cena — 100 tys. dolarów. Ale to już egzotyka.

Następny Infosystem od 10 do 14 kwietnia 1989, także w Poznaniu.

JAROSŁAW DEMINET

Kompilator Ady dla mikroprocesora MC8000

Firma Alsys pracuje intensywnie nad kompilatorami języka Ada przeznaczonymi dla mikroprocesorów Motorola MC68000 i Intel 8086. Kompilatory te mają być dostępne na komputerach opartych na wymienionych procesorach, pracujących pod kontrolą systemu Unix. Do takich właśnie komputerów ma być dostosowany kod wynikowy produkowany przez odpowiedni kompilator. Środowisko programowe Ady musi zawierać bibliotekę Ady, kompilator produkujący efektywny kod wynikowy, program wiążący symboliczny program uruchomieniowy (ang. debugger) oraz wiele innych narzędzi programowych (np. program redagujący). W skład środowiska programowego wchodzi także pakiet wykonawczy, zapewniający funkcje niezbędne do wykonania kodu programu (przetwarzanie równoległe, obsługa sytuacji wyjątkowych) i umożliwiające zainstalowanie kompilatora w danym systemie operacyjnym.

Kompilatory mają służyć konstruowaniu systemów oprogramowania dla komputerów pracujących w sieciach wielokomputerowych. Do implementacji pierwszego z rodziny kompilatorów wybrano środowisko komputera z mikroprocesorem MC68000, pracującym pod kontrolą systemu Unix. W środowisku tym kompilator musi współpracować z podstawowymi narzędziami służącymi do redagowania programów i generowania kodu.

Kod tworzony przez kompilator jest wykonywany w systemie macierzystym. Możliwe więc jest opracowywanie w tym systemie programów aplikacyjnych i narzędziowych, wykorzystujących pełną Adę. Możliwa jest również adaptacja kodu dla innego systemu docelowego z zachowaniem wszystkich właściwości języka oraz pełnej wydajności.

W wypadku wersji kompilatora dla MC68000 wybór Unixa pozwala na uzyskanie przenośności programów w ramach szerokiej klasy komputerów. Architektura kompilatora i jego niezależność maszynowa ułatwia przenoszenie go, po adaptacji zaledwie kilku modułów, dla potrzeb nowego systemu macierzystego.

W dalszej części artykułu przedstawiono opracowywany system narzędzi programowych i jego wersję użytkową dla systemu Unix.

ŚRODOWISKO PROGRAMOWE

W skład środowiska wchodzi oprócz kompilatora wiele programów pomocniczych, niezbędnych do wyprodukowania kodu Ady, oraz programów usługowych pomagających w tworzeniu oprogramowania. Opisane tu narzędzia programistyczne już istnieją lub są w fazie realizacji. Minimalne środowisko programowe języka tworzą: bibliotekarz, program wiążący oraz sam kompilator.

Bibliotekarz pozwala tworzyć bibliotekę Ady, która jest bardzo istotna w systemie opracowywania programów w tym języku. Biblioteka ta zawiera opis każdej jednostki kompilacji: specyfikację pakietu, ciało pakietu, specyfikację i ciało podprogramu. Dla każdej jednostki kompilacji tworzony jest zbiór powiązań i zależności między jednostkami, przechowywane są wszystkie informacje niezbędne do wykonania rozłącznej kompilacji, tzn. opis danych i programów, w postaci pozwalającej na wykorzystywanie ich zarówno przez kompilator jak i przez programy usługowe z jego otoczenia. Bibliotekarz umożliwia przeszukiwanie biblioteki w celu odnalezienia powiązań i zależności między jednostkami oraz wykonania niezbędnych ponownych kompilacji. Pozwala także na przenoszenie programów między użytkownikami (importowanie i eksportowanie) i tworzenie lokalnych wersji jednostek kompilacji oraz tworzenie niezbędnych sprzężeń między poszczególnymi programami usługowymi i kompilatorem.

Program wiążący (ang. binder) tworzy powiązania między kompilatorem i środowiskiem wykonawczym programu lub też między kompilatorem i konsolidatorem systemowym. Spełnia on podwójną rolę. W pierwszym kroku, poszukuje w zbiorach bibliotecznych jednostek kompilacji niezbędnych do wykonania programu, sprawdza ich ważność i określa kolejność ich opracowywania, która może być zadana za pomocą poleceń lub przez pragmy. W drugim kroku, na podstawie danych uzyskanych w pierwszym etapie, program wiążący generuje program opracowywania potrzebnych jednostek, a następnie tworzy plik, w postaci wymaganej przez konsolidator, zawierający kod i dane wytworzone przez kompilator dla każdej wykorzystywanej jednostki. Postać wynikowa po konsolidacji jest użyteczna zwłaszcza w wypadku dołączania programów pisanych w innym języku, za pomocą pragmy INTERFACE.

Do innych narzędzi należą: analizator, program redagujący, edytor tablicy odwołań zewnętrznych i program uruchomieniowy. **Analizator** pozwala na kompletną analizę programu bez konieczności jego pełnej kompilacji. **Program redagujący** służy do redagowania programu w standardowej, czytelnej postaci. **Edytor** tablicy odwołań zewnętrznych pozwala odnaleźć wszystkie wystąpienia danych w określonej jednostce kompilacji oraz wszystkie odwołania do danych i programów należących do danego środowiska programowego.

Program uruchomieniowy jest narzędziem do wykrywania błędów. Używa nazw symbolicznych i pozwala na ustawianie pułapek w czasie wykonywania programu, a także na odczyt lub modyfikację danych. Ponadto, umożliwia wykrywanie i korygowanie błędnego zachowania programów oraz ich profilowanie. Wykrywanie błędów za pomocą programu uruchomieniowego polega na obserwowaniu pętli, miejsc przerwań oraz sytuacji wyjątkowych przy wykonywaniu określonej części kodu. Mechanizm śledzenia pozwala na odnalezienie przyczyn powstania nienormalnej sytuacji. Profilowanie polega na dokonywaniu pomiarów czasu wykonania określonych sekwencji kodu. Szczególną właściwością programu uruchomieniowego jest możliwość stosowania go również dla kodu zoptymalizowanego, przy minimalnej stracie funkcjonalności.

KOMPILACJA

Kompilator jest złożony z dwu głównych części:

- rdzenia (ang. core compiler),
- generatora kodu.

Rdzeń kompilatora jest przystosowany do implementacji w różnych systemach. Jest praktycznie niezależny od typu komputera i może być zaadaptowany do każdego komputera o nowej architekturze, przez wprowadzenie odpowiednich parametrów lub zmianę kilku procedur. Rdzeń składa się z analizatora i ekspandera kodu.

Analizator dokonuje kompletnej analizy syntaktycznej i semantycznej jednostki kompilacji. Dokonuje on identyfikacji danych i podprogramów oraz znajduje w bibliotece wszystkie dane dostępne dla aktualnej jednostki kompilacji (pakiety wywoływane przez klauzulę WITH i jednostkę dołączaną). Analizator wytwarza tablicę symboli oraz drzewo reprezentujące strukturę programu. Informacje te zostają zapamiętane w bibliotece i mogą być wykorzystywane w następnych etapach kompilacji tej jednostki, w czasie kompilacji innych jednostek lub podczas dołączania kolejnych struktur. Postać informacji tworzonych przez analizator jest zbliżona do postaci DIANA, choć możliwe jest też otrzymanie drzewa całkowicie zgodnego z tą postacią. Omawiana implementacja realizuje również częściową optymalizację kodu pośredniego (p. poniżej).

Rola **ekspandera** polega na przetłumaczeniu kodu pośredniego na język bardziej zbliżony do poziomu sprzętu. Tablica symboli i drzewo syntaktyczne są zamieniane na opis obiektów i drzewo opisujące podstawowe operacje na obiektach. W pierwszym kroku, ekspander tworzy implementację typów

Ady i określa sposób alokacji pamięci dla danych i funkcji, na podstawie zawartości pakietu SYSTEM i parametrów implementacji. Następnie, semantyka kodu jest wyrażana za pomocą podstawowych operacji na obiektach, w szczególności całkowicie rozwijane są operacje dostępu do danych. W czasie tworzenia danych pośrednich brane są pod uwagę parametry i specyficzne funkcje konkretnego komputera lub wybranej implementacji. Dotyczą one głównie dostępu do danych.

Opisywany rdzeń kompilatora realizuje ok. 80% procesu kompilacji, współpracując z wieloma możliwymi wersjami ekspandera.

Generator kodu dla MC68000 przyjmuje kod pośredni wytwarzany przez ekspander i tworzy kod binarny programu wraz z blokiem inicjowania danych oraz informację niezbędną dla programu wiążącego. Na działanie generatora składają się trzy przebiegi: alokacja, generowanie kodu i przebieg końcowy. Pierwszy przebieg polega na alokacji danych i konstruowaniu bloków statycznego inicjowania danych. Ostateczna alokacja danych polega na wybraniu danych o stałych rozmiarach, należących do danej przestrzeni adresowej i przypisaniu im pozycji w tej przestrzeni. W czasie generowania kodu, przez rekursywne przeglądanie drzewa pośredniego, tworzonego przez ekspander, wytwarzany jest ciąg instrukcji programu użytkowego wraz z zawartymi w nich adresami danych. Mechanizm alokacji rejestrów i zmiennych tymczasowych pozwala na optymalne zarządzanie zasobami procesora. W przebiegu końcowym wygenerowany kod jest poddawany dodatkowemu przetwarzaniu oraz optymalizacji.

Optymalizacja kodu odbywa się we wszystkich fazach kompilacji. Na poziomie rdzenia kompilatora istnieją dwa przebiegi optymalizacji: jeden z nich dotyczy reprezentacji syntaktycznej przed rozwinięciem kodu, drugi zaś — postaci końcowej. W ostatnim przebiegu optymalizacji przygotowuje się oprócz informacji w drzewie pośrednim, również informacje wykorzystywane przy alokacji rejestrów, umożliwiające wyeliminowanie zbędnych operacji.

Architektura kompilatora umożliwia jego implementację w zredukowanym środowisku sprzętowym. Dostępne obecnie mikrokomputery mają jeszcze wciąż ograniczoną wielkość pamięci operacyjnej i nie są wyposażone w system pamięci wirtualnej ani w wydajny system nakładkowania. Dlatego kompilacja odbywa się w kilku odrębnych przebiegach, co pozwala na ograniczenie rozmiarów kodu: analizator wykonuje trzy przebiegi, ekspander — cztery, a generator kodu — trzy.

Istnienie biblioteki i możliwość przechowywania wszystkich postaci pośrednich pozwala na niezależne wykonywanie poszczególnych faz kompilacji. Z tego też powodu rozmiar wewnętrznych struktur danych kompilatora jest wynaczany w czasie kompilacji, a w razie potrzeby struktury te są implementowane jako obiekty przestrzeni wirtualnej, kontrolowanej przez kompilator.

IMPLEMENTACJA W SYSTEMIE UNIX

Program w Adzie jest wykonywany w systemie Unix jako jeden samodzielny proces, niezależnie od liczby zawartych w nim zadań. Inna implementacja może być realizowana tylko w takiej wersji Unixa, która pozwala na rzeczywiste współdzielenie danych między procesami.

Przestrzeń danych programu w Adzie składa się w systemie Unix z wielu części:

1. **Statyczne dane globalne.** Są to dane zawarte w jednostkach bibliotecznych, będące specyfikacją lub ciałem pakietu, oraz dane generowane przez kompilator, służące do zarządzania jednostkami. Są one implementowane w sposób statyczny. W systemie Unix odpowiadają przestrzeni danych procesu. Z punktu widzenia programu w Adzie, dla każdej jednostki bibliotecznej istnieje oddzielna przestrzeń. Podczas wykonywania kodu jednostki jej dane globalne są dostępne przez wyznaczony rejestr adresowy. Dane innych jednostek są dostępne przez wskaźniki na przestrzeń danych globalnych tych jednostek, umieszczone w przestrzeni danych globalnych jednostki bieżącej. Gdy wykonanie kodu powoduje zmianę bieżącej jednostki bibliotecznej, wtedy następuje również przełączenie kontekstu danych globalnych. Użytkownik może zażądać od kompilatora połączenia wielu przestrzeni danych globalnych, aby uniknąć zmian kontekstu przy przełączaniu między pewnymi jednostkami.

2. **Dynamiczne dane globalne.** Są one również deklarowane w specyfikacjach lub ciałach pakietów, zawartych w jednostkach bibliotecznych, ale ich rozmiar jest ustalany podczas wykonywania programu. Są one alokowane dynamicznie w obszarze stosu systemowego i nie są nigdy usuwane. Dostęp do nich odbywa się za pomocą wskaźnika ustawionego w odpowiedniej statycznej przestrzeni globalnej.

3. **Dane lokalne.** Są to dane deklarowane wewnątrz procedur, alokowane w obszarze stosu danego zadania. Przy wejściu do procedury, na szczycie stosu, rezerwowane jest miejsce dla tych danych, których rozmiar został określony podczas kompilacji. Dane o zmiennych rozmiarach są umieszczane na szczycie stosu w czasie ich opracowania. Dostęp do nich jest możliwy przez wskaźnik umieszczony w przestrzeni statycznej. Odwołania do danych lokalnych są realizowane przez przeznaczony do tego rejestr adresowy, natomiast dostęp do danych poprzednich kontekstów jest realizowany przez wektor odzwierciedlający przebieg zmian kontekstów zadania.

4. **Dane o zmiennym rozmiarze,** np. obiekty typu rekordowego z wariantami. Są alokowane dynamicznie na stercie i dostępne za pomocą wskaźnika lokalnego lub globalnego, jak wszystkie dane dynamiczne. Sterta w systemie Unix jest ograniczona w części dynamicznej przestrzeni danych przez sukcesywny przydział pamięci. Mechanizm przydziału pamięci pozwala na odzyskiwanie wolnych przestrzeni w obszarze sterty.

5. **Obiekty należące do kolekcji.** Są również alokowane na stercie, chyba że kolekcji przypisany jest obszar o rozmiarach określonych przez użytkownika. Wówczas alokacja obiektów należących do tej kolekcji odbywa się w inny, zoptymalizowany sposób.

6. **Zadania.** Implementowane są w ten sam sposób, co obiekty. Dla każdego zadania tworzony jest obiekt zwany blokiem kontrolnym zadania (ang. task control block) oraz oddzielny stos. Zadanie jest rozpoznawane przez adres jego bloku kontrolnego, który może być umieszczony w przestrzeni statycznych danych globalnych, na stosie lub na stercie, zależnie od sposobu zadeklarowania zadania.

Sposób alokacji wszystkich danych może być zmieniony w fazie optymalizacji lub na żądanie użytkownika. Dane dynamiczne, których maksymalny rozmiar jest znany i niewielki, mogą być traktowane jak statyczne. Z kolei dane statyczne dużych rozmiarów są traktowane jak dynamiczne, w celu lepszego wykorzystania przestrzeni danych statycznych, której rozmiar jest ograniczony do 32 lub 64 KB.

Wywoływanie procedur jest realizowane następująco. Parametry są przekazywane przez wartość lub adres — umieszczone na stosie. Na stosie zostaje również umieszczony adres powrotu, a następnie alokowany jest obszar dla statycznych zmiennych lokalnych, przy czym poprzednia wartość wskaźnika stosu zostaje zapamiętana. Zapamiętany zostaje też aktualny kontekst i następuje alokacja danych dynamicznych. Adres wejścia do procedury zostaje przechowany we wskaźniku, w obszarze jej danych globalnych, chyba że procedura ta jest dostępna tylko lokalnie. W chwili wywołania następuje zmiana kontekstu danych globalnych, umożliwiająca procedurze wywołującej określenie adresu procedury wywoływanej i udostępnienie procedurze wywoływanej właściwego kontekstu.

Spotkania (fr. rendez-vous) są implementowane jako podprogramy. Instrukcje **accept** i **entry** są więc implementowane jako podprogramy czasu wykonania z zadaniami, zawierającymi te instrukcje, jako parametrami. Gdy spotkanie jest możliwe, wtedy wywoływany jest odpowiedni kod spotkania. W przeciwnym wypadku, następuje zapamiętanie odpowiednich informacji w bloku kontrolnym zadania, wywołanie instrukcji **entry** zostaje umieszczone w odpowiedniej kolejce oczekiwania, po czym wywoływany jest program szeregujący. Poszczególne punkty synchronizacji są imple-

mentowane również jako wywołania odpowiednich podprogramów.

W mechanizmie obsługi wyjątków silnie wykorzystuje się przetwarzanie tablic. W chwili powstania sytuacji wyjątkowej rozpatruje się szereg możliwych wypadków. Rodzaj wyjątku i sposób jego powstania zostają przekazane do odpowiedniego podprogramu. Jeżeli wyjątek zostaje wykryty przez system lub sprzęt, to jest on przechwycony przez Unix, który określa miejsce i przyczynę jego powstania. Określenie podprogramu w którym powstał wyjątek pozwala procedurze obsługi na wykorzystanie tablic wyjątków podprogramu oraz zawartości stosu do zwolnienia wszystkich zasobów używanych przez podprogram, którego wykonywanie zostaje zaniechane.

W omawianej implementacji wszystkie elementy otoczenia programowego zależne od systemu docelowego zrealizowano jako podprogramy czasu wykonania, dołączane w czasie konsolidacji lub stanowiące zasoby systemowe. Adaptacja zdecydowanej większości oprogramowania nie wymaga więc wprowadzania zmian w źródłowym kodzie programu, lecz polega jedynie na przepisaniu zestawu podprogramów zależnych od systemu.

Oprac. DOROTA INKIELMAN
na podst. BIGRE, 1984

Konferencje

Systemy ekspertowe stan obecny i przyszłość

Ostatnia, doroczna konferencja **International Joint Conference on Artificial Intelligence** — najważniejsza pośród międzynarodowych konferencji poświęconych sztucznej inteligencji — odbyła się w sierpniu 1987 roku, w Mediolanie. Konferencji towarzyszyła wystawa narzędzi sztucznej inteligencji i systemów ekspertowych, w której wzięło udział ponad 60 wystawców z USA i Europy. Uczestnicy konferencji i wystawcy, mając pełny przegląd aktualnego stanu badań i zastosowań systemów ekspertowych na świecie z wyjątkiem Japonii, mogli stwierdzić naocznie, że:

- prawie wszystkie bez wyjątku poważniejsze zastosowania systemów ekspertowych miały jak dotąd miejsce w USA,
- Europa Zachodnia jest daleko w tyle za Stanami Zjednoczonymi, jeśli idzie o rozwój technologii systemów ekspertowych,
- systemy ekspertowe nie są już na ogół niezależnymi programami, lecz coraz częściej stanowią fragment większych systemów.

Amerykanie traktują rozwój technologii systemów ekspertowych bardzo poważnie, inwestując olbrzymie kwoty w budowę dużych systemów. Te firmy, które jako pierwsze zaczęły przed laty konstruować systemy ekspertowe, dysponują już obecnie udanymi zastosowaniami.

Jednym z takich zastosowań, o którym wiele mówiono w czasie konferencji IJCAI '87, był system **ExpertTAX** opracowany przez firmę Coopers and Lybrand. ExpertTAX jest systemem doradczym, wspierającym radą księgowych, prowadzących rejestr wpływów od amerykańskich podatników. Prototyp tego systemu był demonstrowany na różnych konferencjach w 1986 roku. W rok później firma Coopers and Lybrand zakończyła żmudne przechodzenie od prototypowej do użytkowej wersji programu i rozpoczęła jego instalowanie na biurkach swych pracowników. Zainwestowanie dużych kwot pieniędzy w to przedsięwzięcie świadczy o tym, że ExpertTAX uznano za praktyczne i wydajne narzędzie wspomagające pracę księgowych.

Inną wyraźną oznaką handlowego znaczenia i akceptacji systemów ekspertowych było wystąpienie jednego z dyrektorów IBM. Herbert Schorr, szef badań nad sztuczną inteligencją w IBM, stwierdził w swym referacie zatytułowanym „Komercjalizacja sztucznej inteligencji", że IBM posługuje się znaczenia i akceptacji systemów ekspertowymi w pracy administracji oraz w zastosowaniach technicznych, a 75 dalszych systemów znajduje się w budowie. Podając przykłady konkretnych zastosowań w IBM, Schorr podkreślił podstawowe korzyści, jakie płyną z wykorzystania tej technologii:

- łatwy i ciągły dostęp do niezbędnej wiedzy nawet w najodleglejszych oddziałach firmy,
- uwolnienie specjalistów od powtarzalnych czynności i wykorzystanie ich do rozwiązywania mniej rutynowych zadań.

Schorr podkreślił również, że wiele spośród 500 największych firm wymienianych w wykazie Fortune wyszło z fazy badania możliwości nowej technologii i dokonuje obecnie poważnych nwestycji w tej dziedzinie. Niektóre z firm przeznaczają 50% swych wydatków na budowę konwencjonalnego oprogramowania w Pascalu, C lub Cobolu oraz 50% — na budowę systenów ekspertowych. Wyniki prowadzonych prac będą widoczne za kilka lat, a te firmy (lub kraje), które nie inwestują aktualnie w rozwój inteligentnych systemów komputerowych, mogą mieć duże trudności z odrobieniem powstałego opóźnienia.

Konstruowanie konkretnych systemów ekspertowych przez poszczególne firmy to tylko jeden z możliwych rodzajów zastosowań. Wielu prelegentów podkreślało na IJCAJ '87, że wkrótce pojawią się na rynku uniwersalne systemy ekspertowe, rozwiązujące ogólne zadania w takich dziedzinach, jak analiza stanu przedsiębiorstwa, planowanie finansowe z odległym horyzontem czasowym i planowanie przedsięwzięć. Systemy tego rodzaju będą sprzedawane jako gotowe, uniwersalne pakiety programowe i ewentualnie dopasowywane do potrzeb zamawiającego.

Na tle amerykańskich osiągnięć, wykorzystanie systemów ekspertowych w Europie przedstawia się raczej skromnie. Opóźnienie to wynika przede wszystkim z nadmiernej ostrożności w podejmowaniu badań i inwestycji w nowych dziedzinach techniki. Wiele spośród dostępnych obecnie, amerykańskich narzędzi sztucznej inteligencji wywodzi się z prac prowadzonych w latach siedemdziesiątych na amerykańskich uniwersytetach. Europa nie dysponuje podobnym zapleczem badawczym, gdyż w owym czasie prowadzono w Europie niewiele prac naukowych na szerszą skalę.

Brytyjski program Alveya zapełnił w pewnym stopniu lukę w badaniach podstawowych, lecz jego pełna realizacja jest poważnie zagrożona przez znaczne zmniejszenie funduszów. Europejski projekt **ESPRIT**, w którym biorą udział kraje Europy Zachodniej z wyjątkiem Wielkiej Brytanii, poświęca więcej uwagi bezpośredniemu zmaganiu się z konkurencją, niż badaniom podstawowym i nie dostarczył jak dotąd szczególnie wartościowych wyników (być może powodem tego jest fakt, że projekt ESPRIT znalazł się pod zbyt dużym wpływem tzw. „wielkiej dwunastki", czyli 12 największych europejskich firm elektronicznych). Kontynuacją ESPRIT jest obecnie projekt ESPRIT II, który podobnie jak jego poprzednik przeznacza jedynie 4% swego budżetu na badania podstawowe.

Europejscy producenci narzędzi wspomagających budowę systemów ekspertowych nie są na ogół jeszcze przygotowani do podjęcia wyzwania i zainwestowania większych kwot pie-

niędzy. Powszechnie uważa się jednak, że powodem nie jest brak funduszy, lecz brak wyobraźni. W rezultacie prawie wszystkie bez wyjątku systemy ekspertowe, rozwiązujące praktyczne problemy i wyposażone w rozbudowane programy komunikacji człowiek-maszyna, powstały w Stanach Zjednoczonych.

Jakie są charakterystyczne cechy nowych narzędzi wspomagających budowę systemów ekspertowych w porównaniu z dotychczas produkowanymi narzędziami? Przede wszystkim, nowe systemy ekspertowe to nie tylko samodzielne programy realizujące określone funkcje, lecz coraz częściej programy wchodzące w skład większych zastosowań. Najlepsze narzędzia muszą więc współpracować z innymi systemami. Część z tych narzędzi umożliwia dołączenie systemu ekspertowego do nadrzędnego programu obliczeniowego, który uaktywnia go jedynie wtedy, gdy pojawi się potrzeba podjęcia specjalistycznej decyzji (przykładem takiego narzędzia jest **KES** firmy SA&E). Inna grupa nowych narzędzi posiada wbudowane sprzężenia do współpracy z większością popularnych systemów baz danych.

Nowe narzędzia budowy systemów ekspertowych zawierają również bogatsze i elastyczniejsze mechanizmy reprezentacji wiedzy. Coraz powszechniejszym modelem reprezentacji wiedzy staje się model oparty na programowaniu obiektowym, który w istotny sposób rozszerza możliwości prostego modelu regułowego. Wiele firm oferuje w zestawie kilka narzędzi o różnym stopniu zaawansowania:

- prosty i tani (w cenie rzędu kilkuset dolarów) pakiet, który szybko zapoznaje użytkownika z ideą systemu ekspertowego (i oczywiście przywiązuje go do technicznych rozwiązań przyjętych przez daną firmę);
- bardziej złożony pakiet zapewniający współpracę z innymi programami;
- wyrafinowany i drogi (w cenie rzędu kilku lub nawet kilkunastu tysięcy dolarów) pakiet oferujący różne schematy reprezentacji wiedzy.

Dobrym przykładem rodziny narzędzi tego rodzaju jest **Nexus** firmy Human Intellect Systems.

Na wystawie towarzyszącej konferencji IJCAI '87 zademonstrowano szereg nowych narzędzi wspomagających budowę systemów ekspertowych. Pośród nich największe zainteresowanie wywołały trzy pakiety omówione poniżej.

Nexpert Object firmy Neuron Data (444 High Street, Palo Alto, CA 94301, USA) jest przykładem nowoczesnego i elastycznego narzędzia do budowy systemów ekspertowych. Jest ono wyposażone w sprzęg graficzny, pokazujący w przejrzysty sposób relacje między faktami zawartymi w bazie wiedzy. Tworzony system ekspertowy może być częścią większego programu, jak również może współpracować z większością standardowych baz danych i elektronicznych arkuszy obliczeniowych. Nexpert Object jest przeznaczony dla mikrokomputerów Macintosh oraz Vax. Cena mieści się w przedziale od 5 do 10 tysięcy dolarów, w zależności od wersji pakietu.

Instant Expert Plus jest nowym produktem firmy Human Intellect Systems (1670 South Amphlett Boulevard, San Mateo, CA 94402, USA). Różni się od innych narzędzi przede wszystkim tym, że budowany system ekspertowy może dodatkowo posługiwać się danymi graficznymi, przypisując fragmenty obrazu poszczególnym faktom w bazie wiedzy. Przykładowo, użytkownik medycznego systemu diagnostycznego, poproszony o zlokalizowanie bólu, może bezpośrednio wskazać na ekranie odpowiednią część obrazu przedstawiającego ciało ludzkie. Dialog tego rodzaju, tzn. rozszerzony o możliwości graficzne, okazuje się pożyteczny w wielu zastosowaniach. Cena narzędzia wynosi 500 dolarów.

Brytyjska firma Cognitive Applications Ltd. (4 Sillwood Terrace, Brighton, BN1 2LR, Wielka Brytania) przedstawiła na wystawie zintegrowane środowisko programowania zadań sztucznej inteligencji **AlphaPOP**. System ten jest oparty na języku Pop 11, o którym twierdzi się, że jest językiem równie mocnym jak Lisp, lecz pozwalającym na konstruowanie znacznie czytelniejszych programów. Pop 11 był dotychczas dostępny jako język wchodzący w skład kosztownego (kosztującego kilka tysięcy funtów) systemu Poplog. AlphaPOP zrealizowano na mikrokomputerze Macintosh, a jego cena wynosi 450 funtów.

Przewodnim referatem każdej konferencji IJCAI jest wykład zatytułowany „Komputery a myśl ludzka", wygłaszany przez specjalnie zaproszonego badacza sztucznej inteligencji. Wykładowcą zaproszonym na IJCAI '87 był Johan De Kleer z Xerox PARC, który poświęcił swe wystąpienie **systemom wnioskowania jakościowego** (ang. qualitative reasoning). Możliwość rozszerzenia wnioskowania heurystycznego o wnioskowanie jakościowe była jednym z żywiej dyskutowanych tematów tej konferencji.

Odnotowano również nowe osiągnięcia w innych dziedzinach sztucznej inteligencji, takich jak systemy oparte na języku naturalnym, systemy wizyjne i maszynowe uczenie się.

Systemy języka naturalnego — po okresie pewnego zapomnienia — budzą ponownie duże zainteresowanie. Obecnie wykorzystuje się z powodzeniem wcześniejsze prace badawcze w dziedzinie języka naturalnego, budując proste sprzęgi językowe do baz danych, systemów ekspertowych, a nawet gier przygodowych. Znaczenie języka naturalnego wzrośnie jeszcze bardziej, gdy pojawią się praktyczne **systemy z głosowym dostępem do komputera.**

Odnotowuje się również intensywne wysiłki badawcze w dziedzinie wizji komputerowej. **Systemy wizyjne** o ograniczonych możliwościach znajdują już szerokie zastosowanie w zautomatyzowanych fabrykach. Oczekuje się, że bardziej wyrafinowane systemy wizyjne, oparte na „inteligentnym" oprogramowaniu i nowym sprzęcie przetwarzania równoległego znajdą wkrótce liczne zastosowania w przemyśle i poza nim (np. jako automatyczne systemy bezpieczeństwa).

- **Maszynowe uczenie się** jest uważane powszechnie za jeden z ważniejszych (obecnie bardzo aktywnie rozwijanych) obszarów sztucznej inteligencji. Proste techniki indukcji znalazły już zastosowanie w automatycznym generowaniu reguł na podstawie przedłożonych przykładów z danej dziedziny, usprawniając istotnie proces zdobywania wiedzy w systemach ekspertowych. Nowe, aktualnie badane strategie maszynowego uczenia się będą powszechnie wykorzystywane nie tylko w systemach ekspertowych, lecz także w przyszłych systemach wizyjnych, systemach głosowych i innych narzędziach sztucznej inteligencji.

Oprac. M. MACHURA
na podst. Systems International, październik 1987

Komputery szóstej generacji w Japonii

Na uniwersytecie Kumamoto, we współpracy z firmą Omron Teteishi Electronics Company, opracowano pod kierunkiem Takeshi Yamakawy komputer, który może bezpośrednio przetwarzać niezdeterminowane sygnały dźwiękowe. Jest on oparty na nowym typie układów logicznych zwanych rozmytymi (ang. fuzzy), które w przeciwieństwie do dotychczas stosowanych układów binarnych przedstawiają dane według skali przynależności (ułamkowe wartości od 0 do 1). Obejmuje on program zawarty w pamięci przechowującej te ułamkowe wartości, demodulator (zamieniający wartości ułamkowe na binarne) i jednostkę sterującą. Komputer może dokonywać 10 milionów wnioskowań na sekundę. Podstawowymi elementami są tu układy o sprzężeniu emiterowym (ECFL, ang. emitter-coupled fuzzy logic), których cykl pracy jest krótszy od jednej dziesięciomiliardowej części sekundy, oraz układy obcinające, które oddziaływują na funkcje przynależności, pozwalając podzespołom wnioskującym na określanie minimum i maksimum.

Nowy komputer przekształca informację językową na napięcie analogowe w zakresie od 0 do 5 V, które pojawia się na szynie danych. Zapamiętywanie tej informacji dokonuje się jednak w postaci binarnej przy wykorzystaniu zwykłej pamięci operacyjnej, aby ograniczyć zewnętrzne oddziaływanie elektryczne. **(JR)**

Esperanto jako język pośredni w tłumaczeniu komputerowym

„Rozproszone przetwarzanie tekstów" (po esperancku Distribuita Lingvotradukado — DLT) to nazwa długofalowego projektu badawczo-rozwojowego realizowanego przez holenderskie przedsiębiorstwo komputerowe z Utrechtu. Celem projektu jest skonstruowanie komputerowego systemu tłumaczącego dla języków naturalnych.

Według opinii ekspertów całkowicie automatyczne tłumaczenie z jednego języka naturalnego na inny jest obecnie niemożliwe, jeśli w wyniku mamy otrzymać tłumaczenie dobrej jakości. Eksploatowane aktualnie systemy tłumaczące dokonują, co prawda, tłumaczenia automatycznie, lecz wynikowy tekst musi być sprawdzony przez człowieka. Osoba dokonująca korekty musi być tłumaczem i władać zarówno językiem źródłowym, jak i językiem docelowym.

ISTOTA PROJEKTU DLT

Projekt DLT wychodzi z odmiennych założeń. Budowany system tłumaczący ma działać w sieciach komputerowych, gdzie nie ma możliwości dokonania skutecznej, finalnej korekty. Projekt DLT zakłada, że człowiek, współdziałając z komputerem, nie poprawia błędów popełnionych przez system tłumaczący, lecz pomaga mu „zrozumieć" tłumaczony tekst, tak aby tłumaczenie było właściwe. Współpraca ta odbywa się w trybie interakcyjnym, inicjowanym przez system. Ważną cechą tego dialogu jest to, że odbywa się on w języku, z którego tekst jest tłumaczony. Dzięki temu człowiek nie musi być ani tłumaczem, ani znać esperanta, ani języka docelowego.

Projekt DLT jest jednym z dziesięciu największych tego rodzaju projektów na świecie, realizowanych głównie w Europie Zachodniej i w Japonii. Projekt DLT, podobnie jak inne projekty tego rodzaju ma stworzyć system tłumaczący, który będzie praktycznie eksploatowany w latach 1990—2020. Jednak w przeciwieństwie do konkurentów, DLT jest ukierunkowany na tłumaczenie szerokiej gamy języków. Jest pomyślany w ten sposób, aby stosunkowo łatwo można było dodawać kolejne języki, z których — lub na które — dokonuje się tłumaczenia, bez konieczności zmieniania już istniejącego systemu. Podejście to jest atrakcyjne zwłaszcza dla języków spoza grupy indoeuropejskiej i dla języków małych społeczności językowych.

EWOLUCJA PROJEKTU

Jako pierwszy prace nad projektem DLT podjął inżynier Toon Witkam z firmy BSO. Obecnie, wraz z grupą informatyków i lingwistów opracowuje on prototyp systemu dla pary języków: angielski-francuski. Przy dokonywaniu tłumaczenia wykorzystuje się język pośredni, oparty na języku esperanto. Należy wyraźnie podkreślić, że język pośredni projektu DLT nie odpowiada dokładnie esperantu, lecz jest jego zmodyfikowaną wersją, która otrzymała nazwę IL (esp. interlingvo — język pośredni).

Grupa opracowująca DLT dysponuje budżetem 17 milionów guldenów, pochodzących w połowie od samej firmy BSO i w połowie z subwencji holenderskiego Ministerstwa Gospodarki. Projekt DLT ma dostarczyć na rynek konkretny produkt. Aby jednak osiągnąć założony cel, tj. zbudować komputerowy system tłumaczący, konieczne jest wykonanie dużej, teoretyczno-badawczej pracy w wielu dziedzinach lingwistyki i informatyki.

W latach 1985—1986 na bazie przeprowadzonych badań wstępnych rozwijano, początkowo eksperymentalne, a następnie praktycznie następujące elementy systemu tłumaczącego:

- filtr syntaktyczny dla języka IL, który rozpoznaje syntaktycznie poprawne zdania, a odrzuca niepoprawne,
- analizator syntaktyczny dla języka IL, który z poprawnych zdań konstruuje drzewa rozbioru gramatycznego, ze wskaźnikami gramatycznych relacji między poszczególnymi słowami,
- słownik języka IL z cechami syntaktycznymi.

Powyższe elementy służą do tworzenia gramatycznej formy zdań. Jednakże dodatkowym, bardzo ważnym problemem w tłumaczeniach, jest przekazanie dokładnego znaczenia słów, zdań i tekstów. Aby sprostać tym wymaganiom konstruuje się:

- bank wiedzy o słowach, którego funkcja jest podobna do słownika definiującego znaczenia; obecnie bank wiedzy języka IL obejmuje około 50 tysięcy par wyrazów powiązanych relacją wskazującą bezpośredni (możliwy semantycznie) kontakt pojedynczych słów,
- system ekspertowy analizy słów, dokonujący wyboru jednego spośród alternatywnych tłumaczeń pojedynczych słów,
- sformalizowane dwujęzyczne słowniki: IL-angielski i IL-francuski.

W grudniu 1986 i styczniu 1987 roku przeprowadzono testy dokładności analizy słów osiąganej przez system ekspertowy. Badanie prowadził i kontrolował neutralny ekspert, prof. Alan Melby, z amerykańskiego uniwersytetu Grigham-Young. Jego opinia była pozytywna i zachęcająca do dalszych prac.

Następnie rozpoczęto szczegółowe opracowywanie gramatyki podzbioru języka angielskiego, używanej w europejskim przemyśle lotniczym do tworzenia tekstów technicznych. Będzie to pierwszy język źródłowy w projekcie DLT.

BILANS DOTYCHCZASOWYCH OSIĄGNIĘĆ

Spośród dotychczasowych rezultatów, jeden jest szczególnie interesujący. Od 1985 roku do chwili obecnej coraz bardziej jasna stawała się rola sztucznej inteligencji w tłumaczeniach komputerowych. Sztuczna inteligencja stawia sobie za cel badanie komputerowego modelu ludzkiego myślenia. Jej zastosowanie w tłumaczeniach nie jest możliwe bez dysponowania obszernym bankiem wiedzy.

Ważnym osiągnięciem minionych 2—3 lat jest wniosek, że dużą rozrzutnością byłoby tworzenie osobno dla każdego języka banku wiedzy, zbioru reguł wnioskowania, instrukcji porównywania itp.

Esperanto, które według początkowych zamierzeń miało w projekcie DLT pełnić rolę tylko bardzo regularnego i zwięzłego języka pośredniego, w obecnej architekturze projektu odgrywa znacznie ważniejszą rolę. Całe przetwarzanie tłumaczonego tekstu odbywa się w języku IL. IL stał się językiem sztucznej inteligencji. Tekst angielski analizuje się wstępnie tylko na tyle, na ile jest to możliwe za pomocą dostępnych środków formalnych. Gdy w tekście pojawia się niejednoznaczność semantyczna, dla rozwiązania której niezbędne byłoby zrozumienie zawartości tekstu, nie od razu przyjmuje się jedną z dopuszczalnych interpretacji. Najpierw przeprowadza się wszystkie (formalnie dopuszczalne) analizy, a dopiero potem, gdy zdanie to jest już (w wielu wariantach) przetłumaczone na IL, wybiera się za pomocą banku wiedzy i systemu ekspertowego najprawdopodobniejszą wersję. Analiza semantyczna jest procesem tak skomplikowanym i czasochłonnym, że bardziej opłaca się zwiększyć liczbę zdań badanych w jednym języku, niż przetwarzać mniej zdań, za to w większej liczbie języków.

Ta nowa rola języka IL spotęgowała — bardziej niż przewidywano — znaczenie leksykografii, a w ramach niej szczególnie terminografii. Stało się także ważniejsze dokładne zbadanie i zdefiniowanie modyfikacji, które przekształciły esperanto w język IL. Grupa robocza projektu DLT w mar-

cu 1986 rozesłała do interlingwistów, esperantologów i innych zainteresowanych osób ankietę na temat tych modyfikacji. Dzięki tej ankiecie, na którą odpowiedziało pięćdziesięciu językoznawców, a także dzięki własnym doświadczeniom, członkowie grupy DLT mogli określić ostateczną postać języka IL. Wnioskiem najważniejszym jest to, że język IL znacznie zbliżył się do normalnego esperanta. Ograniczenia i modyfikacje zaproponowane w 1983 roku nie są już obecnie konieczne. Stało się to możliwe między innymi dzięki wprowadzeniu niezwykłego modelu syntaktycznego (gramatyka zależności Tesniere'a), który okazał się mocniejszy niż pierwotnie zakładano, a także dzięki lepszemu zrozumieniu tego, że nie jest możliwe dodanie czegokolwiek do systemu (a przecież gramatyka języka jest takim systemem), bez zmieniania funkcji już istniejących składników tego systemu. Tak więc, język IL projektu DLT mocniej opiera się na istniejącej społeczności językowej esperanta. Dotyczy to także częściowo leksykografii: osoba zajmująca się słownikiem musi zaufać swojemu wyczuciu językowemu, uzyskanemu w wyniku posługiwania się danym językiem. To wyczucie językowe zawsze jest esperanckie, a nigdy nie dotyczy języka IL.

W tym sensie język IL, mniej różniący się od esperanta niż można było przypuszczać przed kilkoma laty, nie tylko pozostał, ale nawet umocnił się jako rdzeń całego systemu DLT.

PERSPEKTYWY

Chociaż głównym celem projektu DLT jest opracowanie, do roku 1990, prototypu systemu tłumaczącego z angielskiego na francuski, to jednak już obecnie — choć w bardzo umiarkowanym wymiarze — są prowadzone prace przygotowawcze nad dodatkowymi modułami dla innych języków. Prace te są prowadzone częściowo przy współpracy uniwersytetów lub zespołów badawczych z zainteresowanych krajów. Dotychczas, badania wstępne objęły języki niemiecki i duński, jako języki krajów ze Wspólnoty Europejskiej, oraz fiński i węgierski jako języki nieindoeuropejskie. Rozpoczęto współpracę w zakresie języków romańskich, słowiańskich i wschodnioazjatyckich.

Wydaje się jednak, że ważniejsze niż dodawanie kolejnych języków (z których lub na które się tłumaczy) jest rozwijanie w esperancie specjalnej dziedziny leksykografii — terminografii. Tłumaczenie tekstów fachowych, które mają być przetwarzane przez system docelowy, wymaga wysokiego stopnia dokładności przekładu. Z tego powodu twórcy projektu DLT z zainteresowaniem obserwują obecną aktywność Centrum Terminologicznego (Terminologia Centro) w Budapeszcie, a także prace przy kanadyjskim projekcie Pekoteko.

W dalszej perspektywie projekt DLT mógłby ulec decentralizacji, także i w tym sensie, że różne grupy robocze w różnych miastach i krajach współpracowałyby przy tworzeniu obszernego, wielojęzykowego systemu tłumaczącego, którego jądrem byłoby esperanto.

Oprac. GRABOWICZ
na podst. Esperanto, nr 5, 1987

W skrócie • W skrócie • W skrócie • W skrócie • W skrócie • W skrócie

● Naukowcy Fermi National Accelerator Laboratory w Batavii (stan Illinois, USA) opracowali wyspecjalizowany, równoległy system wieloprocesorowy, który ma możliwości uniwersalnego superkomputera, a wymaga znacznie niższych nakładów finansowych (rzędu 350 tysięcy dolarów w porównaniu z około 10 milionami). Został on zaprojektowany do rozwiązywania problemów fizyki wielkich energii, lecz naukowcy ci współpracują z firmą Omnibyte Corp., z West Chicago, nad przystosowaniem tej techniki do innych zastosowań naukowych, przy dalszym obniżaniu kosztów. Obecnie działający system opiera się na architekturze typu MIMD i zawiera 74 jednoukładowe komputery 32-bitowe. Planowano dodanie 66 takich węzłów, z których każdy ma możliwości obliczeniowe minikomputera VAX 11/780.

*

● Amerykańska firma Advantest America Inc. z Lincolnshire w stanie Illinois opracowała tester T3381 VLSI Test System, któremu można podawać przebiegi testujące na 512 końcówek z częstotliwością 100 MHz, a przez multipleksowanie — podwoić częstotliwość. Jest to czołowe osiągnięcie konstrukcyjne w tej dziedzinie. Większość konkurencyjnych testerów układów VLSI ma nie więcej niż 300 końcówek pomiarowych i częstotliwość poniżej 100 MHz. Cena testera wynosi od 2 do 6 milionów dolarów.

*

● Dyskowe pamięci stałe typu kompaktowego staną się wkrótce istotnym podzespołem komputerów osobistych. Amerykańska filia znanego przedsiębiorstwa japońskiego Hitachi (Hitachi Sales Corp. of America) z Compton w Kalifornii oferuje już dwie wersje stacji takich dysków o pojemności 552 MB. Zastosowano tu technikę podobną jak w gramofonach dźwiękowych tego typu, dzięki czemu oba rodzaje urządzeń są wytwarzane na tym samym oprzyrządowaniu. Wolno stojąca stacja CDR 150S jest sprzedawana po 1000 dolarów, a wersja wbudowana do IBM PC (CDR 2500) — za 940 dolarów. Zestaw obejmuje pakiet interfejsu, kabel dołączający i dysk demonstracyjny.

*

● Firma Zycad Corp. z St. Paul w stanie Minnesota, będąca czołowym wytwórcą sprzętu przyspieszającego symulację układów logicznych i defektów pragnie pokonać swych konkurentów oferując nową, znacznie tańszą serię tego sprzętu, opartą na nowej aparaturze. Nosi ona nazwę Magnus i może bezpośrednio pracować z układami scalonymi utworzonymi przy użyciu popularnych pakietów oprogramowania, jak Cadet firmy HHB Systems. Oznacza to, że użytkownicy nie muszą tworzyć oprogramowania pośredniczącego dla przeprowadzenia symulacji. Komputerem sterującym może być dowolne urządzenie — od skomputeryzowanego stanowiska roboczego, przez minikomputer rodziny VAX, do dużych komputerów IBM.

*

● Motorola Inc., a właściwie dział wyrobów półprzewodnikowych tej firmy (Semiconductor Products Sector) w Phoeni dostarcza za 19 945 dolarów podstawowy system 8-stanowiskowy oparty na magistrali VME. System zawiera procesor MC68020 o częstotliwości zegara 16,67 MHz, koprocesor zmiennoprzecinkowy MC68881, 16 KB bardzo szybkiej pamięci podręcznej (ang. cache) służącej zarówno do przechowywania rozkazów, jak i danych, 2 MB dynamicznej, dwuwejściowej pamięci operacyjnej oraz inteligentny sterownik komunikacyjny i oprogramowanie (System V/68). Pamięć masową tworzą: dysk Winchester o pojemności 70 MB, dysk elastyczny o pojemności 635 KB i zabezpieczająca, strumieniowa jednostka taśmowa. System o oznaczeniu SYS1131UY341 jest dostarczany w kasecie, którą można zamontować w typowym stojaku o szerokości 48,3 cm (19 cali). Zawiera ona 12 stanowisk pakietowych i zasilacz o mocy 400 W.

*

● Wbrew ostatnim pogłoskom, że firmy amerykańskie tracą swą konkurencyjność, badania przeprowadzone przez Stowarzyszenie Wytwórców Sprzętu Komputerowego i Biurowego wykazały, że w ciągu ostatnich pięciu lat (1982—1987) nastąpił wzrost udziału tych przedsiębiorstw w rynku światowym z 35,1% do 35,8%. W okresie tym nastąpił wzrost wartości produkowanego sprzętu z 289,4 mld dolarów do 437,9 mld, przeciętnie o 8,6% rocznie, przy czym silniej następował wzrost wartości sprzętu komputerowego (10,5%), a wolniej telekomunikacyjnego (5,4%).
Krajowa produkcja tego sprzętu wzrastała w tempie 6,9% rocznie, podczas gdy światowe tempo tej produkcji wynosiło 8,5%. Oznacza to niewielki (1,2%) spadek udziału USA w produkcji światowej. Wzrost więc udziału w handlu oznacza znaczne wykorzystanie wyrobów produkowanych poza USA. Zwykle importowane są tanie elementy i podzespoły o dobrej jakości, które wykorzystuje się do produkcji urządzeń następnie eksportowanych. Pozwala to ponadto na wyższe zatrudnienie. Wpływy przedsiębiorstw amerykańskich wzrastają tu o 9,1%, przy przeciętnej światowej 8,6%.

W skrócie • W skrócie • W skrócie • W skrócie • W skrócie • W skrócie

Pliki i zbiory

Właściwe przedstawienie skutków wykonywania operacji wejścia-wyjścia wymaga odróżnienia pojęć **plik** i **zbiór.** W odróżnieniu od zbioru, który jest fizyczną kolekcją danych występującą najczęściej poza programem, plik jest jedynie abstrakcyjną strukturą reprezentującą w programie kolekcję znajdującą się w zbiorze albo dostępną przez urządzenie zewnętrzne.

Możliwość przetwarzania różnych zbiorów za pomocą tego samego pliku stanowi znaczne ułatwienie programowania, ponieważ ogranicza zadanie programującego do koncentrowania się na istotnych elementach algorytmu przetwarzania, bez wnikania w działania specyficzne dla danego systemu operacyjnego oraz w sposób reprezentowania danych w zbiorze lub urządzeniu.

W celu uzasadnienia tezy, że pojęcie pliku nie jest synonimiczne z pojęciem zbioru, wystarczy podać, że np. w języku Turbo Pascal plik składa się wyłącznie z komponentów (zmiennych, tablic albo struktur), natomiast skojarzony z nim zbiór składa się z tych komponentów poprzedzonych preambułą określającą ich liczbę i rozmiary. Jeśli ten argument nie wystarcza, to można dodać, że w języku PL/I, w zależności od obranego trybu otwarcia pliku, może on stanowić interpretację zbioru w odwrotnej kolejności jego rekordów.

Oba te przykłady dobitnie podkreślają istotę różnicy między **obiektem programowym — plikiem**, a **obiektem systemowym — zbiorem.**

JAN BIELECKI

Termin angielski **file** przyjęło się tłumaczyć w polskiej literaturze jako **plik.** Trzeba jednak pamiętać, że w zależności od kontekstu oznacza on albo abstrakcyjną strukturę danych w języku programowania wysokiego poziomu, albo konkretną strukturę danych utworzoną przez system operacyjny w pamięci zewnętrznej. Należy te dwie struktury rozróżniać i dlatego w sytuacjach budzących wątpliwość używa się zwykle dodatkowego określenia **plik logiczny** (dla struktury z poziomu języka) lub **plik fizyczny** (dla struktury z poziomu systemu operacyjnego).

Wydaje się, że tego typu rozróżnienie jest zupełnie wystarczające i nie ma potrzeby wprowadzania nowego terminu, jak to jest proponowane powyżej. Zauważmy ponadto, że zmiana jednego terminu może spowodować reakcję łańcuchową. Jeśli bowiem będziemy mówili i pisali **zbiór** zamiast **plik fizyczny**, to musimy na przykład także zmienić termin **system plików** (ang. file system), stosowany od lat w literaturze dotyczącej systemów operacyjnych, oraz termin **zbiór** (ang. set) używany w Pascalu.

JAN MADEY

Uprzejmie informujemy Czytelników, że egzemplarze INFORMATYKI — bieżące i archiwalne — można kupić nie tylko w kioskach Ruchu. Klubie NOT SIGMY, Zakładzie Kolportażu i Dziale Handlowym (szczegóły podano w WARUNKACH PRENUMERATY), ale również
w lokalu naszej redakcji
ul. Mickiewicza 18. m. 17 w Warszawie, tel. 39-14-34
oraz
w specjalistycznej księgarni PP „Domu Książki"
ul. Mokotowska 51/53 w Warszawie, tel. 28-16-14
Zapraszamy wszystkich zainteresowanych.

W grudniu 1948 roku rozpoczęła pracę grupa Aparatów Matematycznych w Instytucie Matematycznym. Tak więc rok bieżący jest 40. rokiem prac badawczych, wdrożeniowych i produkcyjnych w dziedzinie informatyki w Polsce.

Polskie Towarzystwo Informatyczne przy współudziale Stowarzyszenia Elektryków Polskich organizuje w **dniu 22 października 1988 r. Konferencję Naukową, na której zostaną wygłoszone referaty i przeprowadzone dyskusje, poświęcone pierwszemu dwudziestoleciu informatyki w Polsce.**

Polskie Towarzystwo Informatyczne i Stowarzyszenie Elektryków Polskich wyłoniły Komitet Organizacyjny i Komitet Programowy. Pracami Komitetu Organizacyjnego kierują prof. A. Blikle — przewodniczący i prof. B. Paszkowski — wiceprzewodniczący. Pracami Komitetu Programowego kierują zasłużeni dla polskiej informatyki prof. prof. L. Łukaszewicz i R. Marczyński.

Mamy nadzieję, że dorobek tej konferencji zgromadzi materiały i wyjaśni wiele spraw dotyczących początków informatyki w Polsce oraz stanie się przyczynkiem do większej konsolidacji środowiska informatyków polskich.

Wszystkich zainteresowanych wniesieniem wkładu przy przygotowaniu powyższej konferencji i uczestnictwem w niej prosimy o kontaktowanie się:

Polskie Towarzystwo Informatyczne,
ul. Rzymowskiego 30,
02-697 Warszawa, tel. 43-74-44

Zakłady Tworzyw Sztucznych

„ERG"

w Pustkowie

zatrudnią natychmiast

INŻYNIERÓW INFORMATYKÓW

z praktyką

Przedsiębiorstwo zapewnia mieszkanie,
możliwość osiągania zarobków
od 30 do 50 tys. złotych.

Zgłoszenia przyjmuje
i udziela informacji

Dział Kadr i Szkolenia,

39-206 Pustków 3,
tel. Dębica 40-61 wew. 244, 597, 508.

EO/268/87

Problematyka baz danych

C. J. Wertz: The Data Dictionary — Concepts and Uses. North-Holland, Amsterdam, 1986. ISBN 0-444-70148-6

W książce omówiono pojęcia i problemy występujące przy implementacji słownika danych. Zaprezentowano rozwiązania oraz przedyskutowano właściwości słowników danych. W celu zilustrowania, jak funkcjonują omawiane pojęcia, dokonano przeglądu produktów programowych firm Cullinet, IBM oraz Manager Products. Przedstawiono także prosty, lecz dość typowy projekt słownika. Jego celem jest wyjaśnienie pojęć. Sekwencja rozważań w rozdziale poświęconym temu projektowi odzwierciedla kolejność, w jakiej można napotkać poszczególne problemy podczas planowania i tworzenia słownika danych. Materiał prezentowany w książce może być pomocny do stworzenia planu i słownika dokładnie pasującego do konkretnych potrzeb.

Łącząc doświadczenia praktyczne z teorią, autor zestawił w książce materiał i zagadnienia, które nie są omawiane razem w innych książkach na temat słowników danych i administrowania danymi. Nie jest ona technicznym podręcznikiem opisującym szczegóły działania jakiegokolwiek konkretnego produktu programowego, implementującego słownik danych, i nie jest również pomyślana jako substytut do uważnego czytania technicznych podręczników dostarczanych przez sprzedawców oprogramowania. Jest natomiast uzupełnieniem tej dokumentacji technicznej. Zawiera omówienie pojęć i problemów, które w dokumentach technicznych są często ignorowane lub traktowane w sposób powierzchowny.

Spis treści: Pojęcia i problemy danych podstawowych. Niektóre możliwe rozwiązania. Zintegrowany Słownik danych firmy Cullinet. Słownik danych DB/DC firmy IBM. Zarządca danych. Implementacja prostego słownika danych. Dodatkowe problemy i rozwiązania. Spojrzenie wstecz: Rozszerzenia. Recepta na słownik danych.

B. Musteata, R. Lesser: VSAM Techniques — System Concepts and Programming Procedures. North-Holland, Amsterdam, 1986, ISBN 0-444-70170-2

VSAM (ang. Virtual Storage Access Method) jest metodą dostępu, która jest silnie popierana przez IBM, służącą do projektowania i opracowywania systemów. Jest istotnym narzędziem programowym, którego cechy umożliwiają rozszerzenie systemu i jego konwersję na bazę danych, zapewniając wysoki poziom efektywności działania.

Książka ta pod wieloma względami różni się od innych publikacji na ten temat — zawiera syntezę systemu VSAM, na której może bazować każdy rodzaj wiedzy o nim. Celem tej książki jest zebranie wszystkich pytań, metod i technik, które na każdym etapie użytkowania systemu VSAM przez praktyków mogą reprezentować użyteczną informację. Poruszane w książce zagadnienia obejmują kwestię ofektywności systemu VSAM, relację między systemami CICS a VSAM, procedury odtwarzania i ochrony, definicje obiektów VSAM, procedury przetwarzania, programowanie w VSAM, makrodefinicje w VSAM oraz informacje systemowe. Podane informacje obejmują zagadnienia teoretyczne, jak również rozwiązania praktyczne, przydatne zarówno studentom jak i praktykom.

Spis treści: Dwa modele systemów informatycznych: plikowy i oparty na bazach danych. VSAM, czyli metoda dostępu do pamięci wirtualnej: programy usługowe metody dostępu — AMS i IDCAMS. Programy wsadowe w Cobolu do obsługi plików VSAM. Przetwarzanie indeksów przemiennych (ang. alternate indexes). Definiowanie środowiska VSAM. Inne funkcje usługowe dla metody dostępu. Programowanie dla VSAM-JCL oraz rozważania na temat usuwania defektów (ang. debugging) i efektywności. Procedury zrzutu (ang. backup) i odtwarzania. Zastosowania interakcyjne: CICS i VSAM. Katalogi VSAM-u.

R. Hogan: Diagnostic Techniques for IMS Data Bases. North-Holland, Amsterdam, 1986. ISBN 0-444-70062-5

Dotychczas brakowało skutecznego materiału edukacyjnego na temat praktycznych aspektów diagnostyki i utrzymywania baz danych. Administratorzy baz danych musieli uczyć się przede wszystkim metodą prób i błędów. Ta książka wychodzi naprzeciw zapotrzebowaniu, jest praktycznym przewodnikiem po narzędziach, z których może skorzystać analityk i administrator bazy danych, działającej pod nadzorem systemu Information Management System (IMS). Wyjaśniono w niej, co każde z tych narzędzi potrafi zrobić, kiedy może być pożyteczne, a przede wszystkim — jak wykorzystać wynik jego działania.

W tej książce nie kładzie się nacisku na identyfikowanie i rozwiązywanie konkretnej klasy problemów.. Skoncentrowano się w niej raczej na rozwinięciu podstaw do rozumienia tego, co musi się zdarzyć, aby IMS mógł wykonać swoje zadanie. Dlatego, w wypadku powstania błędu, materiał ten może pomóc w określeniu pytań, które pozwolą zidentyfikować, co się stało i jakie są możliwe rozwiązania. Wszystkie opisane problemy i scenariusze są oparte na rzeczywistym doświadczeniu. Materiał był zbierany i wykorzystywany przez autora w jego własnej diagnostyce rzeczywistych problemów baz danych Zamieszczone w książce liczne przykłady ułatwią czytelnikowi analizę danych, pomogą określić, czego należy szukać i jakie zadać pytania, oraz pomogą znaleźć przyczyny niepożądanego zachowania systemu.

Spis treści: Pojęcia bazy danych w systemie IMS. Zasady zrzutu (ang. backup) i odtwarzania. Narzędzia i pomoce diagnostyczne. Użytkowanie SMU-II. Strojenie bazy danych. Analiza składu (ang. dump), styl bazy danych. Diagnozowanie błędów pointerowych. Podsumowanie.

PROMIS
system zarządzania bazą danych

dokończenie ze s. 3

Do ochrony bazy przed zniszczeniem służą polecenia ABD. Kopię bazy w postaci dogodnej do przechowywania i ładowania można otrzymać za pomocą programu usługowego PRMUNL.

Administrator wywołuje przeznaczone dla niego polecenia ABD w trybie konwersacyjnym z uprzywilejowanego terminala. System steruje dialogiem w języku angielskim. Za pomocą tych poleceń ABD może wykonywać następujące czynności.

- Definiowanie plików — do określania fizycznych parametrów pliku służy polecenie FILE, do określania schematu — polecenie DATA, a podschematu — polecenie VIEW.
- Udostępnienie plików — do otwierania i zamykania plików służy polecenie OPEN.
- Wprowadzanie danych — wstępne ładowanie realizuje polecenie LOAD, następne rekordy danych można dopisywać do pliku poleceniem RLOAD.
- Przechowywanie i odtwarzanie bazy — do kopiowania bazy służy polecenie SAVE, a do kopiowania kroniki — polecenie ARCHIVE. Odtworzenie bazy na podstawie kopii i wprowadzanie do niej zmian według kopii kroniki, realizują odpowiednio polecenia RESTORE i REGENERATE.
- Pobieranie informacji od systemu — informacje o zdefiniowanych plikach i podschematach przekazuje polecenie DIR, a informacje o statusie kanałów dostępu — polecenie TSTAT, umożliwiające także dezaktywowanie kanałów dostępu.

Pozostałe składniki architektury systemu przedstawione na rysunku zostaną omówione w drugiej części artykułu.

Ciechomska H., Wójciekian T., Zaborowska E.: PROMIS — system zarządzania bazą danych (1)

INFORMATYKA 1988, nr 5, s. 1

Pierwsza część charakterystyki systemu zarządzania bazą danych dla minikomputerów klasy SM-4, opracowanego w Instytucie Maszyn Matematycznych w Warszawie.

Ciechomska H., Wójciekian T., Zaborowska E.: PROMIS — a data base management system (1)

INFORMATYKA 1988, No. 5, p. 1

First part of characteristics of the data base management system for SM-4 minicomputers, elaborated in the Institute of Mathematic Machines in Warsaw.

Ciechomska H., Wójciekian T., Zaborowska E.: PROMIS — ein Datenbankverwaltungssystem (1)

INFORMATYKA 1988, Nr. 5, S. 1

Erster Teil einer Charakteristik von Datenbankverwaltungssystem für SM-4--Kleinrechner, der in Institut für Mathematische Maschinen in Warschau entwickelt wurde.

Griswold R. E., Griswold M. T.: Języki wysokiego poziomu do przetwarzania napisów — COMIT, SNOBOL4 i Icon (2)

INFORMATYKA 1988, nr 5, s. 4

Druga część charakterystyki metod i narzędzi do przetwarzania napisów, zawierająca omówienie języka SNOBOL4.

Griswold R. E., Griswold M. T.: High level languages for string processing — COMIT, SNOBOL4 and Icon (2)

INFORMATYKA 1988, No. 5, p. 4

Second part of characteristics of methods and tools for string processing, which contains discussion of the SNOBOL4 language.

Griswold R. E., Griswold M. T.: Höhere Sprachen zur Zeichenfolgenverarbeitung — COMIT, SNOBOL4 und Icon (2)

INFORMATYKA 1988, Nr. 5, S. 4

Zweiter Teil einer Charakteristik von Methoden und Hilfsmittel zur Zeichenfolgenverarbeitung, der eine Besprechung der SNOBOL4-Sprache umfasst.

Sysło M.: Algorytmy kombinatoryczne i ich efektywność (3). Kolorowanie wierzchołków grafu.

INFORMATYKA 1988, nr 5, s. 8

Trzecia część charakterystyki wybranych metod rozwiązywania problemów kombinatorycznych, zawierająca omówienie sposobu rozwiązania problemu kolorowania wierzchołków grafu.

Sysło M.: Combinatorial algorithms and their effectiveness (3). Graph tops colouring

INFORMATYKA 1988, No. 5, p. 8

Third part of selected methods for combinatorial problem solving characteristics, which contains discussion of the method for graph tops colouring problem solution.

Sysło M.: Kombinatorische Algorithmen und ihre Effektivität (3). Graphgipfelfärbung

INFORMATYKA 1988, Nr. 5, S. 8

Dritter Teil einer Charakteristik von ausgewählten Lösungsmethoden für kombinatorische Algorithmen, der eine Besprechung von Lösungsmethode für Graphgipfelfärbung umfasst.

Bartyś M.: Procedury INTR i MSDOS w MS-Pascalu

INFORMATYKA 1988, nr 5, s. 12

Omówienie zasad łączenia i przekazywania parametrów aktualnych między programami napisanymi w językach MASM i MS-Pascal na przykładzie implementacji niestandardowych procedur INTR i MSDOS.

Bartyś M.: INTR and MSDOS procedures in MS-Pascal

INFORMATYKA 1988, No. 5, p. 12

Discussion of rules for merging and actual parameter transmitting between programs written in MASM and MS--Pascal languages on example of INTR and MSDOS non-standard procedures implementation.

Bartyś M.: INTR und MSDOS-Prozeduren in MS-Pascal

INFORMATYKA 1988, Nr. 5, S. 12

Eine Besprechung von Regeln für aktuelle Parametervereinigung und -überweisung zwischen den in MASM und MS-Pascal geschriebenen Programmen auf Beispiel von Implementierung der INTR und MSDOS-nichtstandarden Prozeduren.

Dzik A.: Mazovia 1016 — konfiguracja sprzętowa

INFORMATYKA 1988, nr 5, s. 15

Szczegółowy opis konfiguracji sprzętowej mikrokomputera Mazovia 1016.

Dzik M.: Mazovia 1016 — hardware configuration

INFORMATYKA 1988, No. 5, p. 15

Detailed description of the Mazovia 1016 microcomputer hardware configuration.

Dzik M.: Mazovia 1016 — die Hardwarekonfiguration

INFORMATYKA 1988, Nr. 5, S. 15

Detaillierte Beschreibung der Mazovia 1016-Hardwarekonfiguration.

Bielecki J.: Turbo C. Pliki nagłówkowe, prototypy, deklarowanie funkcji

INFORMATYKA 1988, nr 5, s. 17

Sposoby stosowania plików nagłówkowych i prototypów oraz deklarowania funkcji w języku Turbo C.

Bielecki J.: Turbo C. Header files, prototypes, functions declaring

INFORMATYKA 1988, No. 5, p. 17

Methods for application of header files and prototypes and declaration of functions in the Turbo C language.

Bielecki J.: Turbo C. Vorsatzdateien, Prototypen, Funktionenvereinbarung

INFORMATYKA 1988, Nr. 5, S. 17

Methoden der Anwendung von Vorsatzdateien und Prototypen sowie der Vereinbarung von Funktionen in der Turbo C-Sprache.

Biesiada H., Grzywocz J.: Komputerowe przetwarzanie wyników badań medycznych

INFORMATYKA 1988, nr 5, s. 19

Koncepcja uniwersalnego modularnego sposobu systemu przetwarzania danych medycznych WSAD z wykorzystaniem istniejącego oprogramowania komputera ODRA 1305.

Biesiada H., Grzywocz J.: Computer processing of medical research results

INFORMATYKA 1988, No. 5, p. 19

Conception of the WSAD, a modular universal processing system for medical data with utilization of the ODRA 1305 existing software.

Biesiada H., Grzywocz J.: Computerverarbeitung von Ergebnissen der medizinischen Untersuchungen

INFORMATYKA 1988, Nr. 5, S. 19

Eine Konzeption von WSAD, einem universellen modularen Verarbeitungssystem für medizinische Daten mit Ausnützung der schon bestehenden Software von ODRA 1305-Computer.

6-letnie doświadczenie software'owe sprawdzone w ponad 1000 zakładach pracy

TYLKO **SYSTEM CSK/32** jeśli zarządzanie Twoim przedsiębiorstwem wymaga minikomputera o mocy obliczeniowej Odry oraz kilkunastu skomputeryzowanych stanowisk pracy

System CSK/32 to:

MIKROKOMPUTERY QUATRO CSK/32
— przystosowane do pracy z kilkunastoma terminalami, pracujące w dowolnym trybie graficznym

OPROGRAMOWANIE
— systemowe WDOS, MIKROLAN
— narzędziowe — MEGA BLOK, TABPLAN, PL-TEKST, BGRAF, TRYS
— aplikacyjne — FK K & K, EM K & K, KADRY, PŁACE

WDROŻENIA, SZKOLENIA
— zespoły ds. wdrożeń i szkoleń pracują w:
Gdyni tel. 248 018, tlx 054792 • Krakowie tel. 373 573 • Poznaniu tel. 676 271
Wrocławiu tel. 481 679 • Warszawie tel. 258 528, tlx 816711
Sopocie — w wyspecjalizowanym Salonie Sprzedaży, ul Kombatantów 1

CSK™

computer studio kajkowscy

EO/74/88

7

MIĘDZYWOJEWÓDZKA SPÓŁDZIELNIA PRACY „SIÓDEMKA"

ŁODŹ, AL. KOSCIUSZKI 93,

ZAKŁAD INFORMATYKI I SYSTEMÓW KOMPUTEROWYCH

poleca po **najniższych cenach** w kraju:

— mikrokomputery 8-, 16-, 32-bitowe najwyższej jakości renomowanych firm z całego świata
— urządzenia peryferyjne:
- drukarki — również 24-igłowe i laserowe
- streamery
- napędy dyskowe 3", 5.25"
- monitory monochromatyczne, kolorowe, EGA, HEGA, VGA
- karty rozszerzenia pamięci
- kontrolery
- dyski twarde typu Winchester 20 MB, 40 MB, 60 MB, 80 MB

— systemy wielodostępne i lokalne sieci mikrokomputerowe:
- MULTI-LINK
- D-LINK
- XENIX

— materiały eksploatacyjne:
- dyskietki 5.25" MD2-D
- dyskietki 5.25" MD2-HD
- dyskietki 3" CF2
- dyskietki 3.5" MF 2DD
- taśmy barwiące do wszystkich typów drukarek STAR i NEC

Termin realizacji zamówień **natychmiast** po złożeniu zamówienia. **Bezpłatnie:** szkolenia, kursy, zestawy oprogramowania narzędziowego i użytkowego — przy dostarczeniu kompletnych systemów.

Proponujemy również na wszelkiego rodzaju mikrokomputery programy wspomagające zarządzenie przedsiębiorstwem:

— system finansowo-księgowo-kosztowy
— system zbytu i zaopatrzenia
— system technicznego przygotowania produkcji
— system materiałowy
— system kadrowy i kadrowo-płacowy (również dla pracowników akordowych)
Wymienione systemy pracują w wersjach sieciowych i wielodostępnych.

Wszelkich informacji udzielamy codziennie (oprócz sobót) w siedzibie Zakładu w Łodzi przy Al. Kościuszki 101, tel. 36-51-00, w godzinach 8—16.

EO/1005/87

ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ
Skr. pocztowa 35, 90-955 Łódź 8, tel. 57-25-83

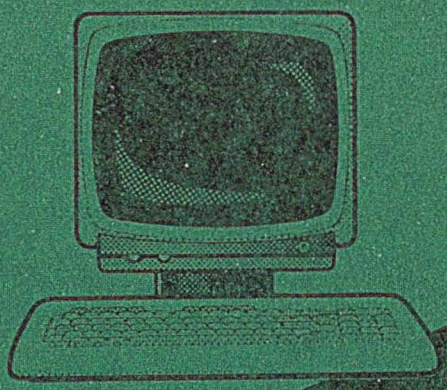
ZEKOM

Oferuje użytkownikom systemów MERA-9150 i REDIFON

TERMINAL EKRANOWY

MR 1241

będący funkcjonalnym odpowiednikiem monitora MERA-7951
Terminal MR 1241 przeznaczony jest do wprowadzania danych do systemu minikomputerowego MERA-9150



JEŚLI INTERESUJE PAŃSTWA:

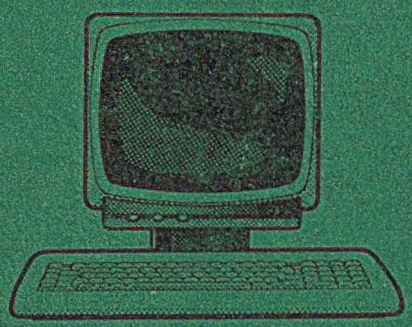
- *profesjonalny sprzęt o wysokiej jakości, niezawodny w eksploatacji*
- *sprawny serwis*
- *krótkie terminy dostaw lub dostawy natychmiastowe*

TO WYROBY ZEKOMU SĄ DO PAŃSTWA DYSPOZYCJI

dla użytkowników systemów MERA-9150 i REDIFON kompatybilny z MERA-7951

EC/1088/88

Dla użytkowników systemów ODRA i ICL



ZEKOM proponuje:

- TERMINAL EKRANOWY MV 2581
  przeznaczony do pracy w systemach ODRA 1300 wyposażonych w grupową jednostkę sterującą JSG-7802 jako odpowiednik monitora MERA-7911N
- TERMINAL EKRANOWY MV 2582E
  przeznaczony do pracy w systemach ODRA 1300 ICL 1900, ICL 2900, ICL System 4 jako odpowiednik monitora 7181/2 firmy ICL
- MULTIPLEKSER TX 82
  przeznaczony do współpracy z terminalami MV 2582E jako 8-kanałowy odpowiednik adaptera QLSA firmy ICL
- ADAPTER LINI TA 42
  przeznaczony do współpracy z terminalami MV 2582E jako 4-kanałowy odpowiednik adaptera LSA firmy ICL

ZEKOM

ZAKŁAD ELEKTRONIKI KOMPUTEROWEJ

Skr. pocztowa 35, 90-955 Łódź 8, tel. 57-25-83